B.Sc. COMPUTER SCIENCE
LAB MANUAL
3rd Semester

Prepared By
Pure and Applied Science Dept.
Computer Science

MIDNAPORE CITY COLLEGE

INSTRUCTIONS TO STUDENTS

- Before entering the lab, the student should carry the following things (MANDATORY)

1. Identity card issued by the college.
2. Class notes
3. Lab observation book
4. Lab Manual
5. Lab Record

- Student must sign in and sign out in the register provided when attending the lab session without fail.
- Come to the laboratory in time. Students, who are late more than 10 min., will not be allowed to attend the lab.
- Students need to maintain 80% attendance in lab if not a strict action will be taken.
- All students must follow a Dress Code while in the laboratory.
- Foods, drinks are NOT allowed.
- All bags must be left at the indicated place.
- Refer to the lab staff if you need any help in using the lab.
- Respect the laboratory and its other users.
- Workspace must be kept clean and tidy after experiment is completed.
- Read the Manual carefully before coming to the laboratory and be sure about what you are supposed to do.
- Do the experiments as per the instructions given in the manual.
- Copy all the programs to observation which are taught in class before attending the lab session.
- Students are not supposed to use floppy disks, pen drives without permission of lab- in charge.
- Lab records need to be submitted on or before the date of submission.

C5P: DATA STRUCTURES LABORATORY MANUAL
(Course: CC-5)

SL. No.	Experiments
1.	Write a program to search an element from a list. Give user the option to perform Linear or Binary search. Use Template functions.
2.	WAP using templates to sort a list of elements. Give user the option to perform sorting using Insertion sort, Bubble sort or Selection sort.
3.	Implement Linked List include functions for insertion, deletion and search of a number, reverse the list and concatenate two linked lists.
4.	Implement Circular Linked List include functions for insertion and deletion.
5.	Perform Stack operations using Linked List implementation.
6.	Perform Stack operations using Array implementation.
7.	Perform Queues operations using Circular Array implementation.
8.	WAP to scan a polynomial using linked list and add two polynomials.
9.	WAP to calculate factorial of a given no. (i) using recursion, (ii) using iteration.
10.	WAP to display Fibonacci series (i)using recursion, (ii) using iteration.
11.	WAP to calculate GCD of 2 number (i) with recursion (ii) without recursion.
12.	WAP to create a Binary Search Tree and include following operations in tree: (a) Insertion (Recursive and Iterative Implementation) (b) Deletion.
13.	WAP to create Binary Tree and display its pre-order, post-order and in-order traversals Recursively.
14.	WAP to create Binary Tree and display its pre-order, post-order and in-order traversals Iteratively.
15.	WAP to implement Diagonal Matrix using one-dimensional array.
16.	WAP to implement Lower Triangular Matrix using one-dimensional array.
17.	WAP to implement Upper Triangular Matrix using one-dimensional array.

1. ***Write a program to search an element from a list. Give user the option to perform Linear or Binary search. Use Template functions.***

Program:

```
#include<iostream>
#include<conio.h>
using namespace std;
//Linear Search Method using Template class
template <class T>
 T Linear_Search(T Iarr[],T key,int n){
    for(int i=0;i<n;i++){
       if(Iarr[i]==key){
          return i;
       }
    }
    return -1;
 }
 //Binary Search Method using Tempalte class
template <class T>
T Binary_Search(T *Iarr,T key,int n)
{
 int l,mid,h;
 l=0;
 h=n-1;
 while(l<=h)
 {
 mid=(l+h)/2;
 if(key==Iarr[mid])
    return mid;
 else if(key<Iarr[mid])
   h=mid-1;
 else
   l=mid+1;
 }
   return -1;
}
int main()
{
//Integer elements
 int Iarr[20],Ielement,i,no;
```

```
cout<<"Enter the number of elements of Integer array: "<<endl;
cin>>no;
for(i=0;i<no;i++){
   cin>>Iarr[i];
}
cout<<"Elements of Integer Array "<<endl;
for(i=0;i<no;i++)
{
 cout<<Iarr[i]<<" ";
}
cout<<"\nEnter an item to be search using Linear_Search Method: ";
cin>>Ielement;
//Linear Search technique is used for Searching the Integer element
int result=Linear_Search(Iarr,Ielement,no);
if(result==-1){
   cout<<"Linear_Search Method Element not found in the list ";
}
else{
   cout<<"Linear_Search Method Element is found at position:
"<<result<<endl;
}
//Binary Search technique is used for Searching the Integer element
cout<<"\nEnter an item to be search using Binary_Search Method: ";
 cin>>Ielement;
result=Binary_Search(Iarr,Ielement,no);
if(result==-1){
   cout<<"Binary_Search Method Element not found in the list "<<endl;
}
else{
   cout<<"Binary_Search Method Element is found at position:
"<<result<<endl;
}
//Float elements
float  F_arr[10],F_element;
cout<<"Enter the "<<no<<" elements of Float array: "<<endl;
for(i=0;i<no;i++){
   cin>>F_arr[i];
}
cout<<"\nElements of Float Array \n";
for(int i=0;i<no;i++)
```

```
 {
  cout<<F_arr[i]<<" ";
 }
 //Linear Search technique is used for Searching the Floating element
 cout<<"\nEnter an item to be search using Linear_Search Method: ";
 cin>>F_element;
 result=Linear_Search(F_arr,F_element,no);
 if(result==-1){
    cout<<"Linear_Search Method Element not found in the list ";
 }
 else{
    cout<<"Linear_Search Method Element is found at position:
"<<result;
 }
 //Binary Search technique is used for Searching the Floating element
 cout<<"\nEnter an item to be search using Binary_Search Method: ";
 cin>>F_element;
result=Binary_Search(F_arr,F_element,no);
if(result==-1){
    cout<<"Binary_Search Method Element not found in the list "<<endl;
 }
 else{
    cout<<"Binary_Search Method Element is found at position:
"<<result<<endl;
 }
//Character elements
char Carr[10]={'a','b','c','d','e','f','g','h','i','j'};
char Celement;
cout<<"Elements of Character Array "<<endl;
 for(i=0;i<10;i++)
 {
  cout<<Carr[i]<<" ";
 }
 //Linear Search technique is used for Searching the Character element
 cout<<"\nEnter an item to be search using Linear_Search Method: ";
 cin>>Celement;
 result=Linear_Search(Carr,Celement,10);
 if(result==-1){
    cout<<"Linear_Search Method Element not found in the list ";
 }
```

```
 else{
    cout<<"Linear_Search Method Element is found at position:
"<<result<<endl;
 }
 //Binary search technique is used for Searching the Character element
 cout<<"\nEnter an item to be search using Binary_Search Method: ";
 cin>>Celement;
result=Binary_Search(Carr,Celement,10);
if(result==-1){
    cout<<"Binary_Search Method Element not found in the list "<<endl;
 }
 else{
    cout<<"Binary_Search Method Element is found at position:
"<<result<<endl;
 }
 getch();
 return 0;
}
```

Input and Output Section:

```
Enter the number of elements of Integer array:
8
10 20 30 40 50 60 70 80
Elements of Integer Array
10 20 30 40 50 60 70 80
Enter an item to be search using Linear_Search Method: 20
Linear_Search Method Element is found at position: 1
Enter an item to be search using Binary_Search Method: 70
Binary_Search Method Element is found at position: 6
Enter the 8 elements of Float array:
1.1 2.3 4.5 6.4 6.55 7.1 7.11 2.1
Elements of Float Array
1.1 2.3 4.5 6.4 6.55 7.1 7.11 2.1
Enter an item to be search using Linear_Search Method: 2.1
Linear_Search Method Element is found at position: 7
Enter an item to be search using Binary_Search Method: 3.7
Binary_Search Method Element not found in the list
Elements of Character Array
a b c d e f g h i j
Enter an item to be search using Linear_Search Method: f
```

```
Linear_Search Method Element is found at position: 5
Enter an item to be search using Binary_Search Method: k
Binary_Search Method Element not found in the list
```

2. ***WAP using templates to sort a list of elements. Give user the option to perform sorting using Insertion sort, Bubble sort or Selection sort.***

Program:

```
#include <iostream>
#include<conio.h>
using namespace std;
template <class T>
void swap(T *x,T *y)
{
   T temp=*x;
   *x=*y;
   *y=temp;
}
//Bubble sort Method using template class
template <class T>
void Bubble(T A[],int n)
{
 int i,j;
 for(i=0;i<n-1;i++)
 {
   for(j=0;j<n-i-1;j++)
   {
      if(A[j]>A[j+1]){
          swap(&A[j],&A[j+1]);
         }
    }
 }
}
//Insersion sort Method using Template class
template <class T>
void Insersion(T A[],int n)
{
 int i,j;
 T x;
```

```
 for(i=1;i<n;i++)
 {
   j=i-1;
   x=A[i];
      while(j>-1 && A[j]>x)
   {
      A[j+1]=A[j];
      j--;
   }
   A[j+1]=x;
 }
}
//Selection Sort Method using Template Class
template <class T>
void SelectionSort(T A[],int n)
{
 int i,j,k;
 for(i=0;i<n-1;i++)
 {
   for(j=k=i;j<n;j++)
   {
      if(A[j]<A[k])
       k=j;
   }
   swap(&A[i],&A[k]);
 }
}

int main()
{

   //Bubble sort technique is used for Integer type array
   int A[100],i,n;
   cout<<"Enter the how many integer elements to be sort using Bubble
sorting:"<<endl;
   cin>>n;
   cout<<"Elements of integer array are: "<<endl;
   for(i=0;i<n;i++){
      cin>>A[i];
   }
```

```
    cout<<"Before Bubble sorting the elements are: "<<endl;
    for(i=0;i<n;i++){
       cout<<A[i]<<" ";
    }
    Bubble(A,n);
    cout<<"\nAfter Bubble sorting the elements are: "<<endl;
    for(i=0;i<n;i++)
    printf("%d ",A[i]);
    printf("\n");
    //Bubble sort technique is used for Floating type array
    float B[100];
    cout<<"Enter the how many float elements to be sort using Bubble
sorting:"<<endl;
    cin>>n;
    cout<<"Elements of floating array are: "<<endl;
    for(i=0;i<n;i++){
       cin>>B[i];
    }
    cout<<"Before Bubble sorting the elements are: "<<endl;
    for(i=0;i<n;i++){
       cout<<B[i]<<" ";
    }
    Bubble(B,n);
    cout<<"\nAfter Bubble sorting the elements are: "<<endl;
    for(i=0;i<n;i++)
    printf("%f ",B[i]);
    printf("\n");

//Insersion sort technique is used for integer type array
    //int A[100],i,n;
    cout<<"Enter the how many integer elements to be sort using Insersion
sorting:"<<endl;
    cin>>n;
    cout<<"Elements of integer array are: "<<endl;
    for(i=0;i<n;i++){
       cin>>A[i];
    }
    cout<<"Before Insersion sorting the elements are: "<<endl;
    for(i=0;i<n;i++){
       cout<<A[i]<<" ";
```

```
    }
    Insersion(A,n);
    cout<<"\nAfter Insersion sorting the elements are: "<<endl;
    for(i=0;i<n;i++)
    printf("%d ",A[i]);
    printf("\n");
    //Insersion sort technique is used for Floating type array
    //float B[100];
    cout<<"Enter the how many float elements to be sort using Insersion
sorting:"<<endl;
    cin>>n;
    cout<<"Elements of floating array are: "<<endl;
    for(i=0;i<n;i++){
       cin>>B[i];
    }
    cout<<"Before Insersion sorting the elements are: "<<endl;
    for(i=0;i<n;i++){
       cout<<B[i]<<" ";
    }
    Insersion(B,n);
    cout<<"\nAfter Insersion sorting the elements are: "<<endl;
    for(i=0;i<n;i++)
    printf("%f ",B[i]);
    printf("\n");

//Selection sort technique is used for integer type array
    //int A[100],i,n;
    cout<<"Enter the how many integer elements to be sort using Selection
sorting:"<<endl;
    cin>>n;
    cout<<"Elements of integer array are: "<<endl;
    for(i=0;i<n;i++){
       cin>>A[i];
    }
    cout<<"Before Selection sorting the elements are: "<<endl;
    for(i=0;i<n;i++){
       cout<<A[i]<<" ";
    }
    SelectionSort(A,n);
    cout<<"\nAfter Selection sorting the elements are: "<<endl;
```

```
    for(i=0;i<n;i++)
    printf("%d ",A[i]);
    printf("\n");
    //Insersion sort technique is used for Floating type array
   // float B[100];
    cout<<"Enter the how many float elements to be sort using Selection
sorting:"<<endl;
    cin>>n;
    cout<<"Elements of floating array are: "<<endl;
    for(i=0;i<n;i++){
       cin>>B[i];
    }
    cout<<"Before Selection sorting the elements are: "<<endl;
    for(i=0;i<n;i++){
       cout<<B[i]<<" ";
    }
    SelectionSort(B,n);
    cout<<"\nAfter Selection sorting the elements are: "<<endl;
    for(i=0;i<n;i++)
    printf("%f ",B[i]);
    printf("\n");
    getch();
    return 0;
}
```

Input and Output Section:

Enter the how many integer elements to be sort using Bubble sorting:
5
Elements of integer array are:
10 23 09 11 15
Before Bubble sorting the elements are:
10 23 9 11 15
After Bubble sorting the elements are:
9 10 11 15 23
Enter the how many float elements to be sort using Bubble sorting:
5
Elements of floating array are:
9.1 2.3 4.5 1.1 2.9
Before Bubble sorting the elements are:
9.1 2.3 4.5 1.1 2.9

After Bubble sorting the elements are:
1.100000 2.300000 2.900000 4.500000 9.100000
Enter the how many integer elements to be sort using Insersion sorting:
6
Elements of integer array are:
19 24 10 11 14
23
Before Insersion sorting the elements are:
19 24 10 11 14 23
After Insersion sorting the elements are:
10 11 14 19 23 24
Enter the how many float elements to be sort using Insersion sorting:
3
Elements of floating array are:
2.3 1.2 .9
Before Insersion sorting the elements are:
2.3 1.2 0.9
After Insersion sorting the elements are:
0.900000 1.200000 2.300000
Enter the how many integer elements to be sort using Selection sorting:
10
Elements of integer array are:
10 20 50 30 21 35 44 9 99 102
Before Selection sorting the elements are:
10 20 50 30 21 35 44 9 99 102
After Selection sorting the elements are:
9 10 20 21 30 35 44 50 99 102
Enter the how many float elements to be sort using Selection sorting:
5
Elements of floating array are:
1.1 .9 .2 .99 1.234
Before Selection sorting the elements are:
1.1 0.9 0.2 0.99 1.234
After Selection sorting the elements are:
0.200000 0.900000 0.990000 1.100000 1.234000

3. ***Implement Linked List include functions for insertion, deletion and search of a number, reverse the list and concatenate two linked lists.***

Insertion Operation using linked list:

Program:

```
/* linked List Insertion */
#include <stdio.h>
#include <stdlib.h>
struct Node
{
 int data;
 struct Node *next;
}*first=NULL;

void create(int A[],int n)
{
 int i;
 struct Node *t,*last;
 first=(struct Node *)malloc(sizeof(struct Node));
 first->data=A[0];
 first->next=NULL;
 last=first;
 for(i=1;i<n;i++)
 {
   t=(struct Node*)malloc(sizeof(struct Node));
   t->data=A[i];
   t->next=NULL;
   last->next=t;
   last=t;
 }
}
int count(struct Node *p)
{
 int l=0;
 while(p)
 {
   l++;
   p=p->next;
 }
```

```
 return l;
}
void Display(struct Node *p)
{
 while(p!=NULL)
 {
   printf("%d ",p->data);
   p=p->next;
 }
}
void Insert(struct Node *p,int index,int x)
{
 struct Node *t;
 int i;

 if(index < 0 || index > count(p))
   return;
 t=(struct Node *)malloc(sizeof(struct Node));
 t->data=x;
 if(index == 0)
 {
   t->next=first;
   first=t;
 }
 else
 {
   for(i=0;i<index-1;i++)
   p=p->next;
   t->next=p->next;
   p->next=t;
 }
}
int main()
{

 //int A[]={10,20,30,40,50};
 //create(A,5);

//Insert Node
 printf("Node Insert: \n");
```

```
 Insert(first,0,5);
 Display(first);
 printf("\nNode Insert: \n");
 Insert(first,1,10);
 Display(first);
 //First Node Insert
 printf("\nFirst Node Insert: \n");
 Insert(first,0,15);
 Display(first);
 //Last Node Insert
 printf("\nLast Node Insert: \n");
 Insert(first,3,20);
 Display(first);
 //Any position Node Insert
 printf("\nGiven position Node Insert: \n");
 Insert(first,2,39);
 Display(first);
 return 0;
}
```

Input and Output Section:

Node Insert:
5
Node Insert:
5 10
First Node Insert:
15 5 10
Last Node Insert:
15 5 10 20
Given position Node Insert:
15 5 39 10 20

Deletion Operation using linked list

Program:

```
/* Linked List Delete element */
#include <stdio.h>
#include <stdlib.h>
struct Node
{
```

```
 int data;
 struct Node *next;
}*first=NULL;
void create(int A[],int n)
{
 int i;
 struct Node *t,*last;
 first=(struct Node *)malloc(sizeof(struct Node));
 first->data=A[0];
 first->next=NULL;
 last=first;
 for(i=1;i<n;i++)
 {
   t=(struct Node*)malloc(sizeof(struct Node));
   t->data=A[i];
   t->next=NULL;
   last->next=t;
   last=t;
 }
}
void Display(struct Node *p)
{
 while(p!=NULL)
 {
   printf("%d ",p->data);
   p=p->next;
 }
}
int count(struct Node *p)
{
 int l=0;
 while(p)
 {
   l++;
   p=p->next;
 }
 return l;
}
int Delete(struct Node *p,int index)
{
```

```
 struct Node *q=NULL;
 int x=-1,i;
 if(index < 1 || index > count(p))
   return -1;
 if(index==1)
 {
   q=first;
   x=first->data;
   first=first->next;
   free(q);
   return x;
 }
 else
 {
   for(i=0;i<index-1;i++)
   {
   q=p;
   p=p->next;
   }
   q->next=p->next;
   x=p->data;
   free(p);
   return x;
 }
}
int main()
{
 int i;
 int A[]={10,20,30,40,50};
 create(A,5);
 printf("Lined List element are: \n");
 Display(first);
 printf("\nDelete element %d\n",Delete(first,2));
 Display(first);
 printf("\nDelete element %d\n",Delete(first,1));
 Display(first);
 printf("\nDelete element %d\n",Delete(first,0));
 Display(first);
return 0;
}
```

Input and Output Section:

Lined List element are:
10 20 30 40 50
Delete element 20
10 30 40 50
Delete element 10
30 40 50
Delete element -1
30 40 50

Search of a number using linked list

Program:

```
/*Searching of a number using Lined List */
#include <stdio.h>
#include <stdlib.h>
struct Node
{
 int data;
 struct Node *next;
}*first=NULL;
void create(int A[],int n)
{
 int i;
 struct Node *t,*last;
 first=(struct Node *)malloc(sizeof(struct Node));
 first->data=A[0];
 first->next=NULL;
 last=first;
 for(i=1;i<n;i++)
 {
   t=(struct Node*)malloc(sizeof(struct Node));
   t->data=A[i];
   t->next=NULL;
   last->next=t;
   last=t;
 }
}
struct Node * LSearch(struct Node *p,int key)
{
```

```
 while(p!=NULL)
 {
    if(key==p->data){
          return p;
    }
    else{
      p=p->next;
    }
 }
   return NULL;
}
struct Node * RSearch(struct Node *p,int key)
{
 if(p==NULL)
   return NULL;
 if(key==p->data)
   return p;
 return RSearch(p->next,key);

}
int main()
{
 struct Node *temp;
 int A[]={3,5,7,10,25,8,32,2},i;
 printf("Linked list elements are: \n");
 for(i=0;i<8;i++){
    printf(" %d ",A[i]);
 }
 //Recursive LinearSearch
 create(A,8);

 temp=RSearch(first,8);
 if(temp)
   printf("\nKey is Found* %d \n",temp->data);
else
   printf("Key is not Found: \n");
//Iterative LinearSearch
temp=LSearch(first,27);
if(temp)
   printf("\nKey is Found* %d \n",temp->data);
```

```
else
    printf("Key is not found: ");

 return 0;
}
```

Input and Output Section:

```
Linked list elements are:
 3  5  7  10  25  8  32  2
Key is Found* 8
Key is not found:
```

Reverse a linked list:

Program:

```
/* Reverse a Linked List */
#include <stdio.h>
#include <stdlib.h>
struct Node
{
 int data;
 struct Node *next;
}*first=NULL,*second=NULL,*third=NULL;
void Display(struct Node *p)
{
 while(p!=NULL)
 {
 printf("%d ",p->data);
 p=p->next;
 }
}
void create(int A[],int n)
{
 int i;
 struct Node *t,*last;
 first=(struct Node *)malloc(sizeof(struct Node));
 first->data=A[0];
 first->next=NULL;
 last=first;
```

```
 for(i=1;i<n;i++)
 {
 t=(struct Node*)malloc(sizeof(struct Node));
 t->data=A[i];
 t->next=NULL;
 last->next=t;
 last=t;
 }
}
int count(struct Node *p)
{
 int l=0;
 while(p)
 {
   l++;
   p=p->next;
 }
 return l;
}
void Reverse1(struct Node *p)
{
 int *A,i=0;
 struct Node *q=p;

 A=(int *)malloc(sizeof(int)*count(p));

 while(q!=NULL)
 {
 A[i]=q->data;
 q=q->next;
 i++;
 }
 q=p;
 i--;
 while(q!=NULL)
 {
 q->data=A[i];
 q=q->next;
 i--;
 }
```

```
}
void Reverse2(struct Node *p)
{
 struct Node *q=NULL,*r=NULL;

 while(p!=NULL)
 {
 r=q;
 q=p;
 p=p->next;
 q->next=r;
 }
 first=q;
}
void Reverse3(struct Node *q,struct Node *p)
{
 if(p)
 {
 Reverse3(p,p->next);
 p->next=q;
 }
 else
 first=q;
}

int main()
{

 int A[]={10,20,40,50,60};
 create(A,5);
 printf("Linked list elements are: \n");
 Display(first);

 Reverse3(NULL,first);
 printf("\nReverse Linked list elements are: \n");
 Display(first);

 return 0;
}
```

Input and Output Section:

Linked list elements are:
10 20 40 50 60
Reverse Linked list elements are:
60 50 40 20 10

Concatenate of two linked list:

Program:

```
/*Concatenate two Linked List*/
#include <stdio.h>
#include <stdlib.h>
struct Node
{
 int data;
 struct Node *next;
}*first=NULL,*second=NULL,*third=NULL;
void Display(struct Node *p)
{
 while(p!=NULL)
 {
 printf("%d ",p->data);
 p=p->next;
 }
}
void create(int A[],int n)
{
 int i;
 struct Node *t,*last;
 first=(struct Node *)malloc(sizeof(struct Node));
 first->data=A[0];
 first->next=NULL;
 last=first;

 for(i=1;i<n;i++)
 {
 t=(struct Node*)malloc(sizeof(struct Node));
 t->data=A[i];
 t->next=NULL;
 last->next=t;
```

```
 last=t;
 }
}
void create2(int A[],int n)
{
 int i;
 struct Node *t,*last;
 second=(struct Node *)malloc(sizeof(struct Node));
 second->data=A[0];
 second->next=NULL;
 last=second;

 for(i=1;i<n;i++)
 {
 t=(struct Node*)malloc(sizeof(struct Node));
 t->data=A[i];
 t->next=NULL;
 last->next=t;
 last=t;
 }
}
/*void Merge(struct Node *p,struct Node *q)
{
 struct Node *last;
 if(p->data < q->data)
 {
 third=last=p;
 p=p->next;
 third->next=NULL;
 }
 else
 {
 third=last=q;
 q=q->next;
 third->next=NULL;
 }
 while(p && q)
 {
 if(p->data < q->data)
 {
```

```
 last->next=p;
 last=p;
 p=p->next;
 last->next=NULL;

 }
 else
 {
 last->next=q;
 last=q;
 q=q->next;
 last->next=NULL;

 }
 }
 if(p)last->next=p;
 if(q)last->next=q;

}
*/
void Concat(struct Node *p,struct Node *q){
   third=p;
   while(p->next!=NULL){
      p=p->next;
   }
   p->next=q;

}
int main()
{

 int A[]={10,20,40,50,60};
 int B[]={15,18,25,30,55};
 create(A,5);
 create2(B,5);


 //Merge(frist,second);
 //Display(third);
 printf("First Linked List \n");
```

```
 Display(first);
 printf("\n");
 printf("Second Linked List \n");
 Display(second);
 printf("\n");
 Concat(first,second);
 printf("Concatenate two Linked List: \n");
 Display(third);
 printf("\n\n");
 return 0;
}
```

Input and Output Section:

First Linked List
10 20 40 50 60
Second Linked List
15 18 25 30 55
Concatenate two Linked List:
10 20 40 50 60 15 18 25 30 55

4. ***Implement Circular Linked List include functions for insertion and deletion.***

Circular Linked List for Insertion Operation:

Program:

```
/*Circular Linked List*/
#include <stdio.h>
#include <stdlib.h>
#include<conio.h>
struct Node
{
 int data;
 struct Node *next;
}*Head;
void create(int A[],int n)
{
 int i;
 struct Node *t,*last;
 Head=(struct Node*)malloc(sizeof(struct Node));
```

```
 Head->data=A[0];
 Head->next=Head;
 last=Head;

 for(i=1;i<n;i++)
 {
   t=(struct Node*)malloc(sizeof(struct Node));
   t->data=A[i];
   t->next=last->next;
   last->next=t;
   last=t;
 }
}
void Display(struct Node *h)
{
 do
 {
   printf("%d ",h->data);
   h=h->next;
 }while(h!=Head);
   printf("\n");
}
/*void RDisplay(struct Node *h)
{
 static int flag=0;
 if(h!=Head || flag==0)
 {
 flag=1;
 printf("%d ",h->data);
 RDisplay(h->next);
 }
 flag=0;
}
*/
int Length(struct Node *p)
{
   int len=0;
 do
 {
   len++;
```

```
    p=p->next;

 }while(p!=Head);
    return len;
}
void Insert(struct Node *p,int index, int x)
{
 struct Node *t;
 int i;
 if(index<0 || index > Length(p))
 return;

 if(index==0)
 {
    t=(struct Node *)malloc(sizeof(struct Node));
    t->data=x;
    if(Head==NULL)
    {
        Head=t;
        Head->next=Head;
    }
 else
 {
    while(p->next!=Head)p=p->next;
    p->next=t;
    t->next=Head;
    Head=t;
 }
 }
 else
 {
    for(i=0;i<index-1;i++)p=p->next;
    t=(struct Node *)malloc(sizeof(struct Node));
    t->data=x;
    t->next=p->next;
    p->next=t;
 }
}

int main()
```

```
{
 int A[]={2,3,4,5,6};
 create(A,5);
 printf("Linked List are: \n");
 Display(Head);

//Insert Node
 printf("\nNode Insert: \n");
 Insert(Head,2,10);
 Display(Head);
 //First Node Insert
 printf("\nFirst Node Insert: \n");
 Insert(Head,0,15);
 Display(Head);
 //Last Node Insert
 printf("\nLast Node Insert: \n");
 Insert(Head,7,20);
 Display(Head);
 //Any position Node Insert
 printf("\nGiven position Node Insert: \n");
 Insert(Head,2,39);
 Display(Head);
getch();
return 0;
}
```

Input and Output Section:

Linked List are:
2 3 4 5 6

Node Insert:
2 3 10 4 5 6

First Node Insert:
15 2 3 10 4 5 6

Last Node Insert:
15 2 3 10 4 5 6 20

Given position Node Insert:

15 2 39 3 10 4 5 6 20

Circular Linked List for Deletion Operation:

Program:

```
/*Circular Linked List Deletion*/
#include <stdio.h>
#include <stdlib.h>
#include<conio.h>
struct Node
{
 int data;
 struct Node *next;
}*Head;
void create(int A[],int n)
{
 int i;
 struct Node *t,*last;
 Head=(struct Node*)malloc(sizeof(struct Node));
 Head->data=A[0];
 Head->next=Head;
 last=Head;
 for(i=1;i<n;i++)
 {
    t=(struct Node*)malloc(sizeof(struct Node));
    t->data=A[i];
    t->next=last->next;
    last->next=t;
    last=t;
 }
}
void Display(struct Node *h)
{
 do
 {
    printf("%d ",h->data);
    h=h->next;
 }while(h!=Head);
    printf("\n");
}
```

```
int Length(struct Node *p)
{
   int len=0;
 do
 {
   len++;
   p=p->next;

 }while(p!=Head);
   return len;
}

int Delete(struct Node *p,int index)
{
   struct Node *q;
   int i,x;

 if(index <0 || index >Length(Head))
   return -1;
 if(index==1)
 {
   while(p->next!=Head)p=p->next;
   x=Head->data;
 if(Head==p)
 {
   free(Head);
   Head=NULL;
 }
 else
 {
   p->next=Head->next;
   free(Head);
   Head=p->next;
   }
 }
 else
 {
   for(i=0;i<index-2;i++)
   p=p->next;
   q=p->next;
```

```
    p->next=q->next;
    x=q->data;
    free(q);
  }
  return x;
}
int main()
{
  int A[]={2,3,4,5,6,7,8,9,10};
  create(A,9);
  printf("Linked List are: \n");
  Display(Head);

  //First Node Delete
  printf("\nFirst Node Delete: \n");
  Delete(Head,1);
  Display(Head);

  //Last Node Delete
  printf("\nLast Node Delete: \n");
  Delete(Head,8);
  Display(Head);

  //Any position Node Insert
  printf("\nGiven position Node Delete: \n");
  Delete(Head,5);
  Display(Head);
  getch();
return 0;
}
```

Input and Output Section:

Linked List are:
2 3 4 5 6 7 8 9 10

First Node Delete:
3 4 5 6 7 8 9 10

Last Node Delete:
3 4 5 6 7 8 9

Given position Node Delete:
3 4 5 6 8 9

5. Perform Stack operations using Linked List implementation.

Program:

```
/*Stack using Linked List*/
#include <stdio.h>
#include <stdlib.h>
struct Node
{
 int data;
 struct Node *next;
}*top=NULL;

void push(int x)
{
 struct Node *t;
 t=(struct Node*)malloc(sizeof(struct Node));
 if(t==NULL)
    printf("stack is full\n");
 else
 {
    t->data=x;
    t->next=top;
    top=t;
 }
}

int pop()
{
 struct Node *t;
 int x=-1;
 if(top==NULL)
    printf("Stack is Empty\n");
 else
 {
    t=top;
    top=top->next;
```

```
    x=t->data;
    free(t);
  }
  return x;
}

void Display()
{
  struct Node *p;
  p=top;
  while(p!=NULL)
  {
    printf("%d ",p->data);
    p=p->next;
  }
  printf("\n");
}
int main()
{
// push the elements
printf("Push the elements are: \n");
//last elements top of the stack
  push(10);
  push(20);
  push(30);
  push(40);
  push(50);
  Display();

  printf("pop element %d \n",pop());
  printf("pop element %d %d ",pop(),pop()); //right to left operation
  return 0;
}
```

Input and Output Section:

Push the elements are:
50 40 30 20 10
pop element 50
pop element 30 40

6. Perform Stack operations using Array implementation.

Program:

```
/* Stack using Array */
#include <stdio.h>
#include <stdlib.h>
struct Stack
{
 int size;
 int top;
 int *S;
};
void create(struct Stack *st)
{
 printf("Enter Size ");
 scanf("%d",&st->size);
 st->top=-1;
 st->S=(int *)malloc(st->size*sizeof(int));
}
void Display(struct Stack st)
{
 int i;
 for(i=st.top;i>=0;i--)
    printf("%d ",st.S[i]);
 printf("\n");

}
void push(struct Stack *st,int x)
{
 if(st->top==st->size-1)
    printf("Stack overflow\n");
 else
 {
    st->top++;
    st->S[st->top]=x;
 }
}
int pop(struct Stack *st)
{
    int x=-1;
```

```
    if(st->top==-1)
        printf("Stack Underflow\n");
    else
    {
        x=st->S[st->top--];
    }
 return x;
}

int peek(struct Stack st,int index)
{
 int x=-1;
 if(st.top-index+1<0)
    printf("Invalid Index \n");
    x=st.S[st.top-index+1];
 return x;
}
int isEmpty(struct Stack st)
{
 if(st.top==-1)
    return 1;
 return 0;
}
int isFull(struct Stack st)
{
    return st.top==st.size-1;
}
int stackTop(struct Stack st)
{
    if(!isEmpty(st))
         return st.S[st.top];
 return -1;
}
int main()
{
 struct Stack st;
 create(&st);
 //push the elements into the Stack
 printf("Stack elements are: \n");
 push(&st,10);
```

```
push(&st,20);
push(&st,30);
push(&st,40);
push(&st,50);
push(&st,60);
//Stack overflow when Enter the size 5
Display(st);
//pop the elements from a Stack
printf("pop element: %d \n",pop(&st));
printf("After the pop operation Stack elements are: \n");
Display(st);
//Stack Underflow when more then 6 elements pop out
//printf("pop element: %d %d %d %d %d
%d\n",pop(&st),pop(&st),pop(&st),pop(&st),pop(&st),pop(&st));

//Peek the element into the Stack
printf("peek element: %d \n",peek(st,1));
printf("After the peek operation Stack elements are: \n");
Display(st);
return 0;
}
```

Input and Output Section:

Enter Size 6
Stack elements are:
60 50 40 30 20 10
pop element: 60
After the pop operation Stack elements are:
50 40 30 20 10
peek element: 50
After the peek operation Stack elements are:
50 40 30 20 10

Enter Size 5
Stack elements are:
Stack overflow
50 40 30 20 10
pop element: 50
After the pop operation Stack elements are:
40 30 20 10

```
peek element: 40
After the peek operation Stack elements are:
40 30 20 10

Enter Size 6
Stack elements are:
60 50 40 30 20 10
pop element: 60
After the pop operation Stack elements are:
50 40 30 20 10
Stack Underflow
pop element: -1 10 20 30 40 50
Invalid Index
peek element: 0
After the peek operation Stack elements are:
```

7. Perform Queues operations using Circular Array implementation.

Program:

```
/* Queue Operation Using Circular Array Implementation*/
#include <stdio.h>
#include <stdlib.h>
struct Queue
{
 int size;
 int front;
 int rear;
 int *Q;
};
void create(struct Queue *q,int size)
{
 q->size=size;
 q->front=q->rear=0;
 q->Q=(int *)malloc(q->size*sizeof(int));
}
void enqueue(struct Queue *q,int x)
{
 if((q->rear+1)%q->size==q->front)
 printf("Queue is Full\n");
 else
```

```
 {
 q->rear=(q->rear+1)%q->size;
 q->Q[q->rear]=x;
 }
}
int dequeue(struct Queue *q)
{
 int x=-1;

 if(q->front==q->rear)
 printf("Queue is Empty\n");
 else
 {
 q->front=(q->front+1)%q->size;
 x=q->Q[q->front];
 }
 return x;
}
void Display(struct Queue q)
{
 int i=q.front+1;

 do
 {

 printf("%d ",q.Q[i]);
 i=(i+1)%q.size;
 }while(i!=(q.rear+1)%q.size);

 printf("\n");
}
int main()
{
 struct Queue q;
 create(&q,6);
 printf("\nQueue elements are: \n");
 //enqueue Operation
 enqueue(&q,10);
 enqueue(&q,20);
 enqueue(&q,30);
```

```
enqueue(&q,40);
enqueue(&q,50);
//enqueue(&q,60);
Display(q);
//dequeue Operation
printf("dequeue element: %d \n",dequeue(&q));
printf("After dequeue Queue elements are: \n");
Display(q);
return 0;
}
```

Input and Output Section:

Queue elements are:
10 20 30 40 50
dequeue element: 10
After dequeue Queue elements are:
20 30 40 50

8. WAP to scan a polynomial using linked list and add two polynomials.

Program:

```
//Polynomial addition using linked list
# include <stdio.h>
# include <stdlib.h>
struct node
{
        float coef;
        int expo;
        struct node *link;
};

 int display(struct node *ptr)
{
        if(ptr==NULL)
        {
                printf("Empty\n");
                return 0;
        }
        while(ptr!=NULL)
        {
```

```
                printf("(%.1fx^%d) + ", ptr->coef,ptr->expo);
                ptr=ptr->link;
        }
        printf("\b\b \n"); // \b\b to erase the last + sign
}
struct node *insert(struct node *start,float co,int ex)
{
        struct node *ptr,*tmp;
        tmp= (struct node*)malloc(sizeof(struct node));
        tmp->coef=co;
        tmp->expo=ex;

        //list empty or exp greater than first one
        if(start==NULL || ex>start->expo)
        {
                tmp->link=start;
                start=tmp;
        }
        else
        {
                ptr=start;
                while(ptr->link!=NULL && ptr->link->expo>ex)
                        ptr=ptr->link;
                tmp->link=ptr->link;
                ptr->link=tmp;
                if(ptr->link==NULL)  //item to be added in the end
                        tmp->link=NULL;
        }
        return start;
}
struct node *enter(struct node *start)
{
        int i,n,ex;
        float co;
        printf("How many terms you want to enter : ");
        scanf("%d",&n);
        printf("Enter each term with coeff and exp\n");
        for(i=1;i<=n;i++)
        {
                scanf("%f %d",&co,&ex);
```

```
                start=insert(start,co,ex);
        }
        return start;
}
struct node *poly_add(struct node *p1,struct node *p2)
{
        struct node *p3_start,*p3,*tmp;
        p3_start=NULL;
        if(p1==NULL && p2==NULL)
                return p3_start;

        while(p1!=NULL && p2!=NULL )
        {
                tmp= (struct node*)malloc(sizeof(struct node));
                if(p3_start==NULL)
                {
                        p3_start=tmp;
                        p3=p3_start;
                }
                else
                {
                        p3->link=tmp;
                        p3=p3->link;
                }
                if(p1->expo > p2->expo)
                {
                        tmp->coef=p1->coef;
                        tmp->expo=p1->expo;
                        p1=p1->link;
                }
                else
                        if(p2->expo > p1->expo)
                        {
                                tmp->coef=p2->coef;
                                tmp->expo=p2->expo;
                                p2=p2->link;
                        }
                        else
                                if(p1->expo == p2->expo)
                                {
```

```
                                        tmp->coef=p1->coef + p2->coef;
                                        tmp->expo=p1->expo;
                                        p1=p1->link;
                                        p2=p2->link;
                                }
        }
        while(p1!=NULL)
        {
                tmp= (struct node*)malloc(sizeof(struct node));
                tmp->coef=p1->coef;
                tmp->expo=p1->expo;
                if (p3_start==NULL) /*poly 2 is empty*/
                {
                        p3_start=tmp;
                        p3=p3_start;
                }
                else
                {
                        p3->link=tmp;
                        p3=p3->link;
                }
                p1=p1->link;
        }
        while(p2!=NULL)
        {
                tmp= (struct node*)malloc(sizeof(struct node));
                tmp->coef=p2->coef;
                tmp->expo=p2->expo;
                if (p3_start==NULL) /*poly 1 is empty*/
                {
                        p3_start=tmp;
                        p3=p3_start;
                }
                else
                {
                        p3->link=tmp;
                        p3=p3->link;
                }
                p2=p2->link;
}
```

```
        p3->link=NULL;
        return p3_start;
}
int main( )
{
        struct node *p1_start,*p2_start,*p3_start;
        p1_start=NULL;
        p2_start=NULL;
        p3_start=NULL;
        printf("Polynomial 1 :\n");
        p1_start=enter(p1_start);
        printf("Polynomial 2 :\n");
        p2_start=enter(p2_start);
        p3_start=poly_add(p1_start,p2_start);
        printf("Polynomial 1 is :  ");
        display(p1_start);
        printf("Polynomial 2 is :  ");
        display(p2_start);
        printf("Added polynomial is :  ");
        display(p3_start);
        return 0;
}
```

Input and Output Section:

```
Polynomial 1 :
How many terms you want to enter : 3
Enter each term with coeff and exp
1.2 5
2.5 4
4 3
Polynomial 2 :
How many terms you want to enter : 3
Enter each term with coeff and exp
2 6
1 2
1 0
Polynomial 1 is :  (1.2x^5) + (2.5x^4) + (4.0x^3)
Polynomial 2 is :  (2.0x^6) + (1.0x^2) + (1.0x^0)
Added polynomial is :  (2.0x^6) + (1.2x^5) + (2.5x^4) + (4.0x^3) + (1.0x^2)
+ (1.0x^0)
```

9. WAP to calculate factorial of a given no. (i) using recursion, (ii) using iteration.

Program:

```
#include <stdio.h>
#include<conio.h>
//Iterative function definition
long ifact(int n){
 long fact=1,i;
 for(i=1;i<=n;i++){
     fact=fact*i;
 }
 return fact;
}
//Recursion function definition
long rfact(int n){
    //base condition
    if(n<=1){
        return n;
    }
    // Recrsive Procedure
    else{
        return n*rfact(n-1);
    }
}
int main()
{
    int n;
    printf("Enter the number \n");
    scanf("%d",&n);
    //iterative function call
    printf("Iteration Method: %d Factorial is %ld",n,ifact(n));
    //recursion function call
    printf("\nRecursion Method: %d Factorial is %ld",n,rfact(n));
    getch();
    return 0;
}
```

Input and Output Section:

```
Enter the number
5
Iteration Method: 5 Factorial is 120
Recursion Method: 5 Factorial is 120
Enter the number
9
Iteration Method: 9 Factorial is 362880
Recursion Method: 9 Factorial is 362880
```

10. WAP to display Fibonacci series (i)using recursion, (ii) using iteration.

Program:

```
#include <stdio.h>
#include<conio.h>
//Iterative function definition
void ifib(int n){
   int a=0,b=1,c,i;
   if(n<1){
      printf("Fibonacci series : %d ",a);
   }
   else if(n==1){
      printf("Fibonacci series : %d %d ",a,b);
            }
      else{
         printf("Fibonacci series are: %d %d",a,b);
         for(i=2;i<=n;i++){
            c=a+b;
            a=b;
            b=c;

         printf(" %d",c);
         }
      }
   }
//Recursion function definition
int rfib(int n){
   int a=0,b=1;
//base condition
   if(n<=1){
```

```
        return n;
    }
//Recursive Procedure
    else{
        return rfib(n-2)+rfib(n-1);
    }
}
int main()
{
    int n,result;
    printf("Enter the number \n");
    scanf("%d",&n);
    //iterative function call
    ifib(n);
    //recursion function call
    result=rfib(n);
    printf("\nLast value of Fibonacci series %d",result);
    getch();
    return 0;
}
```

Input and Output Section:

```
Enter the number
13
Fibonacci series are: 0 1 1 2 3 5 8 13 21 34 55 89 144 233
Last value of Fibonacci series 233

Enter the number
10
Fibonacci series are: 0 1 1 2 3 5 8 13 21 34 55
Last value of Fibonacci series 55
```

11. WAP to calculate GCD of 2 number (i) with recursion (ii) without recursion.

Program:

```
#include <stdio.h>
#include<conio.h>
//Iterative function definition
int igcd(int n,int m){
```

```
    while(m!=n){
        if(m>n){
            m=m-n;
        }
        else{
            n=n-m;
        }
            }
return m;
    }
//Recursion function definition
int rgcd(int n,int m){
    //base condition
    if(n==m){
        return n;
    }
    // Recrsive Procedure
    if(m>n){
        return rgcd(m-n,n);
    }
        rgcd(m,n-m);
}
int main()
{
    int a,b;
    printf("Enter the two number \n");
    scanf("%d%d",&a,&b);
    //iterative function call
    printf("Iterative Method %d and %d GCD is %d ",a,b,igcd(a,b));
    //recursion function call
    printf("\nRecursion Method %d and %d GCD is %d",a,b,rgcd(a,b));
    getch();
    return 0;
}
```

Input and Output Section:

Enter the two number
13 117
Iterative Method 13 and 117 GCD is 13
Recursion Method 13 and 117 GCD is 13

```
Enter the two number
93 131
Iterative Method 93 and 131 GCD is 1
Recursion Method 93 and 131 GCD is 1
```

12. WAP to create a Binary Search Tree and include following operations in tree: (a) Insertion (Recursive and Iterative Implementation) (b) Deletion.

(a) Binary Search Tree Insertion Operation (Recursive and Iterative Implementation):

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
struct Node{
   struct Node *lchild;
   int data;
   struct Node *rchild;
}*root=NULL;
void Insert(int key)
{
   struct Node *t=root;
   struct Node *r,*p;
   if(root==NULL){
      p=(struct Node *)malloc(sizeof(struct Node));
      p->data=key;
      p->lchild=p->rchild=NULL;
      root=p;
      return ;
   }
   while(t!=NULL){
      r=t;
      if(key<t->data)
         t=t->lchild;
      else if(key>t->data)
         t=t->rchild;
      else
         return;
```

```
    }
        p=(struct Node *)malloc(sizeof(struct Node));
        p->data=key;
        p->lchild=p->rchild=NULL;
        if(key<r->data) r->lchild=p;
        else r->rchild=p;
}
    void Inorder(struct Node *p){
        if(p){
            Inorder(p->lchild);
            printf("%d ",p->data);
            Inorder(p->rchild);
        }
    }
struct Node * Search(int key){
    struct Node *t=root;
    while(t!=NULL){
        if(key==t->data)
            return t;
        else if(key<t->data)
            t=t->lchild;
        else
            t=t->rchild;
    }
    return NULL;
}

struct Node *RInsert(struct Node *p,int key){
    struct Node *t=NULL;
    if(p==NULL){
        t=(struct Node *)malloc(sizeof(struct Node));
        t->data=key;
        t->lchild=t->rchild=NULL;
        return t;
    }
        if(key<p->data)
            p->lchild=RInsert(p->lchild,key);
        else if(key>p->data)
            p->rchild=RInsert(p->rchild,key);
```

```
        return p;
}

int main(){
    struct Node *temp;
    printf("Iterative BST: \n");
    Insert(10);
    Insert(5);
    Insert(20);
    Insert(8);
    Insert(30);
    Inorder(root);
    printf("\n");

     temp=Search(30);
    if(temp!=NULL){
        printf("Element %d is Found \n",temp->data);
    }
    else
        printf("Element is not found \n");

    printf("Recursive BST: \n");
    root=RInsert(root,10);
    RInsert(root,5);
    RInsert(root,20);
    RInsert(root,8);
    RInsert(root,30);
    Inorder(root);
    printf("\n");
    temp=Search(20);
    if(temp!=NULL){
        printf("Element %d is Found \n",temp->data);
    }
    else
        printf("Element is not found \n");
    getch();
    return 0;
}
```

Input and Output Section:

Iterative BST:
5 8 10 20 30
Element 30 is Found
Recursive BST:
5 8 10 20 30
Element 20 is Found

(b) Binary Search Tree Deletion Operation (Recursive and Iterative Implementation):

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
struct Node{
   struct Node *lchild;
   int data;
   struct Node *rchild;
}*root=NULL;
void Insert(int key)
{
   struct Node *t=root;
   struct Node *r,*p;
   if(root==NULL){
      p=(struct Node *)malloc(sizeof(struct Node));
      p->data=key;
      p->lchild=p->rchild=NULL;
      root=p;
      return ;
   }
   while(t!=NULL){
      r=t;
      if(key<t->data)
         t=t->lchild;
      else if(key>t->data)
         t=t->rchild;
      else
         return;
   }
```

```
        p=(struct Node *)malloc(sizeof(struct Node));
        p->data=key;
        p->lchild=p->rchild=NULL;
        if(key<r->data) r->lchild=p;
        else r->rchild=p;
}
    void Inorder(struct Node *p){
        if(p){
            Inorder(p->lchild);
            printf("%d ",p->data);
            Inorder(p->rchild);
        }
    }
/*struct Node * Search(int key){
    struct Node *t=root;
    while(t!=NULL){
        if(key==t->data)
            return t;
        else if(key<t->data)
            t=t->lchild;
        else
            t=t->rchild;
    }
    return NULL;
}
*/
struct Node *RInsert(struct Node *p,int key){
    struct Node *t=NULL;
    if(p==NULL){
        t=(struct Node *)malloc(sizeof(struct Node));
        t->data=key;
        t->lchild=t->rchild=NULL;
        return t;
    }
        if(key<p->data)
            p->lchild=RInsert(p->lchild,key);
        else if(key>p->data)
            p->rchild=RInsert(p->rchild,key);

        return p;
```

```
}
int Height(struct Node *p){
   int x,y;
   if(p==NULL) return 0;
   x=Height(p->lchild);
   y=Height(p->rchild);
   return x>y?x+1:y+1;
}
struct Node *InSucc(struct Node *p){
   while(p && p->lchild!=NULL)
      p=p->lchild;
   return p;
}
struct Node *InPre(struct Node *p){
   while(p && p->rchild!=NULL)
      p=p->rchild;
   return p;
}
//BST Delete from a Element
struct Node *Delete(struct Node *p,int key){
   struct Node *q;
   if(p==NULL)
      return NULL;
   if(p->lchild==NULL && p->rchild==NULL){
      if(p==root)
         root=NULL;
      free(p);
         return NULL;
   }
   if(key<p->data)
      p->lchild=Delete(p->lchild,key);
   else if(key>p->data)
      p->rchild=Delete(p->rchild,key);
   else
      {
         if(Height(p->lchild)>Height(p->rchild)){
            q=InPre(p->lchild);
            p->data=q->data;
            p->lchild=Delete(p->lchild,q->data);
         }
```

```
            else{
                q=InSucc(p->rchild);
                p->data=q->data;
                p->rchild=Delete(p->rchild,q->data);
            }
        }
        return p;
}

int main(){

    printf("Iterative BST: \n");
    Insert(10);
    Insert(5);
    Insert(20);
    Insert(8);
    Insert(30);
    Inorder(root);
    printf("\nAfter Iterative BST Delete Element:  ");
    Delete(root,5);
    Inorder(root);
    printf("\n");


    printf("Recursive BST: \n");
    root=RInsert(root,10);
    RInsert(root,5);
    RInsert(root,20);
    RInsert(root,8);
    RInsert(root,30);
    Inorder(root);
    Delete(root,20);
    printf("\nAfter Recursive BST Delete Element:  ");
    Inorder(root);
    printf("\n");
    getch();
    return 0;
}
```

Input and Output Section:

Iterative BST:
5 8 10 20 30
After Iterative BST Delete Element: 8 10 20 30
Recursive BST:
5 8 10 20 30
After Recursive BST Delete Element: 5 8 10 30

13. WAP to create Binary Tree and display its pre-order, post-order and in-order traversals Recursively.

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include "Queue.h"
#include<conio.h>
struct Node *root=NULL;
void Treecreate()
{
 struct Node *p,*t;
 int x;
 struct Queue q;
 create(&q,100);

 printf("Enter root value ");
 scanf("%d",&x);
 root=(struct Node *)malloc(sizeof(struct Node));
 root->data=x;
 root->lchild=root->rchild=NULL;
 enqueue(&q,root);

 while(!isEmpty(q))
 {
 p=dequeue(&q);
 printf("enter left child of %d ",p->data);
 scanf("%d",&x);
 if(x!=-1)
 {
 t=(struct Node *)malloc(sizeof(struct Node));
 t->data=x;
```

```
 t->lchild=t->rchild=NULL;
 p->lchild=t;
 enqueue(&q,t);
 }
 printf("enter right child of %d ",p->data);
 scanf("%d",&x);
 if(x!=-1)
 {
 t=(struct Node *)malloc(sizeof(struct
Node));
 t->data=x;
 t->lchild=t->rchild=NULL;
 p->rchild=t;
 enqueue(&q,t);
 }
 }
}
void Preorder(struct Node *p)
{
 if(p)
 {
 printf("%d ",p->data);
 Preorder(p->lchild);
 Preorder(p->rchild);
 }
}
void Inorder(struct Node *p)
{
 if(p)
 {
 Inorder(p->lchild);
 printf("%d ",p->data);
 Inorder(p->rchild);
 }
}
void Postorder(struct Node *p)
{
 if(p)
 {
 Postorder(p->lchild);
```

```
 Postorder(p->rchild);
 printf("%d ",p->data);
 }
}
int main()
{
 Treecreate();
 printf(" \nIn order ");
 Inorder(root);
 printf(" \nPre order ");
 Preorder(root);
 printf("\nPost Order ");
 Postorder(root);
 getch();
 return 0;
}
```

Queue Header File

```
#include<stdio.h>
#include<stdlib.h>
struct Node
{
 struct Node *lchild;
 int data;
 struct Node *rchild;
};
struct Queue
{
 int size;
 int front;
 int rear;
 struct Node **Q;
};
void create(struct Queue *q,int size)
{
 q->size=size;
 q->front=q->rear=0;
 q->Q=(struct Node **)malloc(q->size*sizeof(struct
Node *));
}
```

```
void enqueue(struct Queue *q,struct Node *x)
{
 if((q->rear+1)%q->size==q->front)
 printf("Queue is Full");
 else
 {
 q->rear=(q->rear+1)%q->size;
 q->Q[q->rear]=x;
 }
}
struct Node * dequeue(struct Queue *q)
{
 struct Node* x=NULL;

 if(q->front==q->rear)
 printf("Queue is Empty\n");
 else
 {
 q->front=(q->front+1)%q->size;
 x=q->Q[q->front];
 }
 return x;
}
int isEmpty(struct Queue q)
{
 return q.front==q.rear;
}
```

Input and Output Section:

```
Enter root value 10
enter left child of 10 20
enter right child of 10 30
enter left child of 20 40
enter right child of 20 50
enter left child of 30 60
enter right child of 30 70
enter left child of 40 80
enter right child of 40 90
enter left child of 50 -1
```

```
enter right child of 50 -1
enter left child of 60 -1
enter right child of 60 -1
enter left child of 70 -1
enter right child of 70 -1
enter left child of 80 -1
enter right child of 80 -1
enter left child of 90 -1
enter right child of 90 -1

In order 80 40 90 20 50 10 60 30 70
Pre order 10 20 40 80 90 50 30 60 70
Post Order 80 90 40 50 20 60 70 30 10
```

14. WAP to create Binary Tree and display its pre-order, post-order and in-order traversals Iteratively.

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"
#include "Queue.h"
#include<conio.h>
struct Node *root=NULL;
void Treecreate()
{
 struct Node *p,*t;
 int x;
 struct Queue q;
 create(&q,100);

 printf("Enter root value ");
 scanf("%d",&x);
 root=(struct Node *)malloc(sizeof(struct Node));
 root->data=x;
 root->lchild=root->rchild=NULL;
 enqueue(&q,root);

 while(!isEmpty(q))
 {
```

```
 p=dequeue(&q);
 printf("enter left child of %d ",p->data);
 scanf("%d",&x);
 if(x!=-1)
 {
 t=(struct Node *)malloc(sizeof(struct Node));
 t->data=x;
 t->lchild=t->rchild=NULL;
 p->lchild=t;
 enqueue(&q,t);
 }
 printf("enter right child of %d ",p->data);
 scanf("%d",&x);
 if(x!=-1)
 {
 t=(struct Node *)malloc(sizeof(struct
Node));
 t->data=x;
 t->lchild=t->rchild=NULL;
 p->rchild=t;
 enqueue(&q,t);
 }
 }
}
void IPreorder(struct Node *p){
        struct Stack stk;
        Stackcreate(&stk,100);
        while(p || !isEmpty(stk))
        {
                if(p){
                        printf("%d ",p->data);
                        push(&stk,p);
                        p=p->lchild;
                }
                else{
                        p=pop(&stk);
                        p=p->rchild;
                }
        }
}
```

```
void IInorder(struct Node *p){
        struct Stack stk;
        Stackcreate(&stk,100);
        while(p || !isEmpty(stk))
        {
                if(p){

                        push(&stk,p);
                        p=p->lchild;
                }
                else{
                        p=pop(&stk);
                        printf("%d ",p->data);
                        p=p->rchild;
                }
        }
}

int main()
{
 Treecreate();
 printf(" \nIn order ");
 IInorder(root);
 printf(" \nPre order ");
 IPreorder(root);

 getch();
 return 0;
}
```

Stack Header File

```
#include <stdio.h>
#include <stdlib.h>
//#include "Queue.h"
struct Stack
{
 int size;
 int top;
 struct Node **S;
};
```

```
void Stackcreate(struct Stack *st,int size)
{
 st->size=size;
 st->top=-1;
 st->S=(struct Node **)malloc(st->size*sizeof(struct Node *));
}

void push(struct Stack *st,struct Node *x)
{
 if(st->top==st->size-1)
 printf("Stack overflow\n");
 else
 {
 st->top++;
 st->S[st->top]=x;
 }

}
struct Node *pop(struct Stack *st)
{
 struct Node *x=NULL;
 if(st->top==-1)
 printf("Stack Underflow\n");
 else
 {
 x=st->S[st->top--];
 }
 return x;
}

int isEmpty(struct Stack st)
{
 if(st.top==-1)
          return 1;
 return 0;
}
int isFull(struct Stack st)
{
 return st.top==st.size-1;
}
```

Input and Output Section:

```
Enter root value 10
enter left child of 10 20
enter right child of 10 30
enter left child of 20 40
enter right child of 20 50
enter left child of 30 60
enter right child of 30 70
enter left child of 40 -1
enter right child of 40 -1
enter left child of 50 -1
enter right child of 50 -1
enter left child of 60 -1
enter right child of 60 -1
enter left child of 70 -1
enter right child of 70 -1

In order 40 20 50 10 60 30 70
Pre order 10 20 40 50 30 60 70
```

15. WAP to implement Diagonal Matrix using one-dimensional array.

Program:

```
#include<stdio.h>
#include<conio.h>
struct Matrix
{
 int A[10];
 int n;
};
void Set(struct Matrix *m,int i,int j,int x)
{
 if(i==j)
 m->A[i-1]=x;

}
int Get(struct Matrix m,int i,int j)
{
 if(i==j)
 return m.A[i-1];
```

```
 else
 return 0;
}
void Display(struct Matrix m)
{
 int i,j;
 for(i=0;i<m.n;i++)
 {
   for(j=0;j<m.n;j++)
   {
   if(i==j)
       printf("%d ",m.A[i]);
   else
       printf("0 ");
 }
   printf("\n");
 }
}
int main()
{
//clrscr();
 struct Matrix m;
 m.n=4;

 Set(&m,1,1,5);
 Set(&m,2,2,8);
 Set(&m,3,3,9);
 Set(&m,4,4,12);
// printf("%d \n",Get(m,3,3));
 Display(m);
 getch();
 return 0;
}
```

Input and Output section:

```
5 0 0 0
0 8 0 0
0 0 9 0
0 0 0 12
```

16. WAP to implement Lower Triangular Matrix using one-dimensional array.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct Matrix
{
 int *A;
 int n;
};
void Set(struct Matrix *m,int i,int j,int x)
{
 if(i>=j)
 m->A[m->n*(j-1)+(j-2)*(j-1)/2+i-j]=x;
}
int Get(struct Matrix m,int i,int j)
{
 if(i>=j)
 return m.A[m.n*(j-1)+(j-2)*(j-1)/2+i-j];
 else
 return 0;
}
void Display(struct Matrix m)
{
 int i,j;
 for(i=1;i<=m.n;i++)
 {
 for(j=1;j<=m.n;j++)
 {
 if(i>=j)
 printf("%d  ",m.A[m.n*(j-1)+(j-2)*(j-1)/2+i-j]);
 else
```

```
 printf("0   ");
 }
 printf("\n");
 }
}
int main()
{
 struct Matrix m;
 int i,j,x;
//clrscr();
 printf("Enter Dimension");
 scanf("%d",&m.n);
 m.A=(int *)malloc(m.n*(m.n+1)/2*sizeof(int));
 printf("enter all elements");
 for(i=1;i<=m.n;i++)
 {
 for(j=1;j<=m.n;j++)
 {
 scanf("%d",&x);
 Set(&m,i,j,x);
 }
 }
 printf("\n\n");
 Display(m);

getch();
 return 0;
}
```

Input and Output Section:

```
Enter Dimension
4
enter all elements
10 20 30 40
11 22 33 44
12 24 36 48
13 26 39 52

Lower Triangular Matrix are:
10   0    0   0
```

```
11  22   0    0
12  24  36   0
13  26  39  52
```

17. WAP to implement Upper Triangular Matrix using one-dimensional array.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct Matrix
{
 int *A;
 int n;
};
void Set(struct Matrix *m,int i,int j,int x)
{
 if(i>=j)
 m->A[m->n*(j-1)+(j-2)*(j-1)/2+i-j]=x;
}
int Get(struct Matrix m,int i,int j)
{
 if(i>=j)
 return m.A[m.n*(j-1)+(j-2)*(j-1)/2+i-j];
 else
 return 0;
}
void Display(struct Matrix m)
{
 int i,j;
 for(i=1;i<=m.n;i++)
 {
 for(j=1;j<=m.n;j++)
 {
 if(i<=j)
 printf("%d  ",m.A[m.n*(i-1)+(i-2)*(i-1)/2+j-i]);
 else
 printf("0   ");
 }
```

```
 printf("\n");
 }
}
int main()
{
 struct Matrix m;
 int i,j,x;
//clrscr();
 printf("Enter Dimension");
 scanf("%d",&m.n);
 m.A=(int *)malloc(m.n*(m.n+1)/2*sizeof(int));
 printf("enter all elements");
 for(i=1;i<=m.n;i++)
 {
 for(j=1;j<=m.n;j++)
 {
 scanf("%d",&x);
 Set(&m,i,j,x);
 }
 }
 printf("\n");
 printf("Upper Triangular Matrix are: \n");
 Display(m);

getch();
 return 0;
}
```

Input and Output Section:

```
10 20 33 44
12 11 23 45
12 34 56 78
13 23 45 23

Upper Triangular Matrix are:
10  12  12  13
0    11  34  23
0    0    56  45
0    0    0    23
```

C6P: OPERATING SYSTEMS LABORATORY MANUAL
(Course: CC-6)

SL.NO.	Experiments
1.	Write a program (using fork () and/or exec () commands) where parent and child execute: a) same program, same code. b) same program, different code. c) before terminating, the parent waits for the child to finish its task.
2.	Write a program to report behaviour of Linux kernel including kernel version, CPU type and model. (CPU information)
3.	Write a program to report behaviour of Linux kernel including information on configured memory, amount of free and used memory (memory information).
4.	Write a program to print file details including owner access permissions, file access time, where file name is given as argument.
5.	Write a program to copy files using system calls.
6.	Write program to implement FCFS scheduling algorithm.
7.	Write program to implement Round Robin scheduling algorithm.
8.	Write program to implement SJF scheduling algorithm.
9.	Write program to calculate sum of n numbers using thread library.
10.	Write a program to implement first-fit, best-fit and worst-fit allocation strategies

1. ***Write a program (using fork () and/or exec () commands) where parent and child execute:***

a) same program, same code.

Program:

sanjoy@SANJUVAI:~$ gedit same.c

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main(){
int pid_t,pid,p;
p=fork();
pid=getpid();
if(p<0){
        fprintf(stderr,"Frok failded");
return 1;
}
printf("Output of fork id: %d \n",p);
printf("process id is : %d \n",pid);
return 0;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ gcc same.c -o sh.out
sanjoy@SANJUVAI:~$ ./sh.out
Output of fork id: 564
process id is : 563
Output of fork id: 0
process id is : 564
sanjoy@SANJUVAI:~$
```

b) same program, different code.

Program:

```
sanjoy@SANJUVAI:~$ gedit different.c
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main(){
int pid;
pid=fork();
if(pid<0){
printf("\n Error ");
exit(1);
}
 else if(pid==0){
   printf("\n Hello I am child process ");
   printf("\n my pid is %d \n",getpid());
   exit(0);
 }
  else{
  printf("\n Hello i am parent process  ");
  printf("\n My actual pid is %d \n",getpid());
  exit(1);
  }
return 0;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ gcc different.c -o sh.out
sanjoy@SANJUVAI:~$ ./sh.out

 Hello i am parent process
 My actual pid is 714

 Hello I am child process
 my pid is 715
```

c) before terminating, the parent waits for the child to finish its task.

Program:

```
sanjoy@SANJUVAI:~$ gedit parentwait.c
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main(){
int pid;
pid=fork();
if(pid<0){
printf("\n Error ");
exit(1);
}
 else if(pid==0){
   printf("\n Hello I am child process ");
   printf("\n my pid is %d \n",getpid());
   exit(0);
 }
  else if(pid>0){
  printf("\n Hello i am parent process  ");
  printf("\n My actual pid is %d \n",getpid());
  //wait(NULL);
  exit(1);
  }
return 0;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ gcc parentwait.c -o sh.out
sanjoy@SANJUVAI:~$ ./sh.out

 Hello i am parent process
 My actual pid is 907

 Hello I am child process
 my pid is 908
```

2. Write a program to report behaviour of Linux kernel including kernel version, CPU type and model. (CPU information)

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/utsname.h>
using namespace std;
int main()
{
   int m=0;
   struct utsname s1;
   m=uname(&s1);

   if(m==0)
   {
      printf("\n The name of System:%s , system" , s1.sysname);
      printf("\n The version:%s" , s1.version);
      printf("\n The Machine:%s" , s1.machine);
      printf("\n");
      system("cat /proc/cpuinfo | awk 'NR==3, NR==4{print}' \n");
   }

   else
   {
      printf("Error");
   }
   return 0;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ touch question2.cpp
sanjoy@SANJUVAI:~$ g++ question2.cpp
sanjoy@SANJUVAI:~$ ./a.out

 The name of System:Linux , system
 The version:#1 SMP Wed Nov 23 01:01:46 UTC 2022
```

The Machine:x86_64

cpu family : 6

model : 126

3. ***Write a program to report behaviour of Linux kernel including information on configured memory, amount of free and used memory (memory information).***

Program:

```
#include<stdlib.h>
#include<iostream>
using namespace std;
int main()
{
cout<<"\nThe kernel version is, \n";
system("cat /proc/sys/kernel/osrelease") ;
cout<<"\nThe CPU space: \n";
system("cat /proc/cpuinfo | awk 'NR==3, NR==4{print}' \n ");
cout<<"\nAmount of cpu time since system was last booted is: ";
system("cat /proc/uptime \n");
cout<<"\nThe configured memory is : \n";
system("cat /proc/meminfo | awk 'NR==1, NR==4{print $2}' \n ");
//cout<<"\nAmount of free memory: \n";
//system("cat /proc/meminfo |awk 'NR = 2{Print $2}' \n ");
cout<<"Amount of used memory is: \n";
system("cat /proc/meminfo | awk '{ if (NR==1) a=$2; if(NR==2) b=$2
}END {print a-b} ' \n");
return 0;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ g++ memory.cpp
sanjoy@SANJUVAI:~$ ./a.out

The kernel version is,
5.15.79.1-microsoft-standard-WSL2

The CPU space:
```

```
cpu family      : 6
model           : 126

3531.89 28231.67
Amount of cpu time since system was last booted is:
The configured memory is :
3880936
3286912
3380688
11896
Amount of used memory is:
594024
```

4. ***Write a program to print file details including owner access permissions, file access time, where file name is given as argument.***

Program:

```
#include<iostream>
using namespace std;
#include<stdlib.h>
#include<stdio.h>
#include<sys/stat.h>
#include<sys/types.h>
#include<unistd.h>
int main(int argc, char *argv[])
{
int i;
struct stat s;
if (argc < 2)
{
cout<<"\n enter filename in";
//exit();
}
for(i=1;i<argc; i++)
{
cout<<"File : "<<argv[i]<<"\n";
if(stat(argv[i],&s)<0)
cout<<"error in obtaining stats In";
else
{
```

```
cout<<"owner UID : "; cout<<s.st_uid; cout<<"\n";
cout<<"group ID :"; cout<<s.st_gid; cout<<"\n";
cout<<"Access permissions : "; cout<<s.st_mode; cout<<"\n";
cout<<"Access Time :" ;cout<<s.st_atime; cout<<"\n";
cout<<"File Size : "; cout<<s.st_size; cout<<"\n";
cout<<"File Size(in blocks) : "; cout<<s.st_blksize; cout<<"\n";
}
}
return 0;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ cat>f1
Well, Come to Midnapore City College.
^Z
[3]+  Stopped                 cat > f1
sanjoy@SANJUVAI:~$ g++ test2.cpp
sanjoy@SANJUVAI:~$ ./a.out f1
File : f1
owner UID : 1000
group ID :1000
Access permissions : 33188
Access Time :1674936817
File Size : 37
File Size(in blocks) : 4096
```

5. Write a program to copy files using system calls.

Program:

```
sanjoy@SANJUVAI:~$ gedit copyfile.c

#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
int main(int argc, char *argv[])
 {
   int f1, f2;
   char buff[50];
   long int n;
```

```
 if(((f1 = open(argv[1], O_RDONLY)) == -1 || ((f2=open(argv[2],
O_CREAT |
 O_WRONLY | O_TRUNC, 0700))== 1)))
   {
     perror("problem in file");
     exit(1);
   }

 while((n=read(f1, buff, 50))>0)

    if(write(f2, buff, n)!=n)
      {
        perror("problem in writing");
        exit(3);
      }

    if(n==-1)
      {
        perror("problem in reading");
        exit(2);
      }

     close(f2);
     exit(0);
 }
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ cc copyfile.c
sanjoy@SANJUVAI:~$ cat>bsc.txt
This is BSc Computer Science File.
BSc Computer Science File is copy into BCA File.
^Z
[7]+  Stopped                 cat > bsc.txt
sanjoy@SANJUVAI:~$ ./a.out bsc.txt bca.txt
sanjoy@SANJUVAI:~$ cat bca.txt
This is BSc Computer Science File.
BSc Computer Science File is copy into BCA File.
```

6. Write program to implement FCFS scheduling algorithm.

Program:

```
// C program for implementation of FCFS scheduling
#include<stdio.h>
// Function to find the waiting time for all processes
void findWaitingTime(int processes[], int n, int bt[], int wt[])
{
    // waiting time for first process is 0
    wt[0] = 0;

    // calculating waiting time
    for (int i = 1; i < n ; i++ )
            wt[i] = bt[i-1] + wt[i-1] ;
}

// Function to calculate turn around time
void findTurnAroundTime( int processes[], int n, int bt[], int wt[], int
tat[])
{
    // calculating turnaround time by adding bt[i] + wt[i]
    for (int i = 0; i < n ; i++)
            tat[i] = bt[i] + wt[i];
}
//Function to calculate average time
void findavgTime( int processes[], int n, int bt[])
{
    int wt[n], tat[n], total_wt = 0, total_tat = 0;
    //Function to find waiting time of all processes
    findWaitingTime(processes, n, bt, wt);
    //Function to find turn around time for all processes
    findTurnAroundTime(processes, n, bt, wt, tat);
    //Display processes along with all details
    printf("Processes Burst time Waiting time Turn around time\n");
    // Calculate total waiting time and total turn around time
    for (int i=0; i<n; i++)
    {
            total_wt = total_wt + wt[i];
            total_tat = total_tat + tat[i];
            printf(" %d ",(i+1));
```

```
            printf("\t   %d", bt[i] );
            printf("\t\t %d",wt[i] );
            printf("\t\t %d\n",tat[i] );
        }
        int s=(float)total_wt / (float)n;
        int t=(float)total_tat / (float)n;
        printf("Average waiting time = %d",s);
        printf("\n");
        printf("Average turn around time = %d ",t);
}
int main()
{
    //process id's
    int processes[] = { 1, 2, 3};
    int n = sizeof processes / sizeof processes[0];

    //Burst time of all processes
    int burst_time[] = {10, 5, 8};

    findavgTime(processes, n, burst_time);
    return 0;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ gcc fcfs.c -o a.out
sanjoy@SANJUVAI:~$ ./a.out
Processes Burst time Waiting time Turn around time
 1        10          0            10
 2        5           10           15
 3        8           15           23
Average waiting time = 8
Average turn around time = 16
```

7. ***Write program to implement Round Robin scheduling algorithm.***

Program:

```
#include<stdio.h>
//#include<conio.h>
int main()
{
```

```
    // initlialize the variable name
    int i, NOP, sum=0,count=0, y, quant, wt=0, tat=0, at[10], bt[10],
temp[10];
    float avg_wt, avg_tat;
    printf(" Total number of process in the system: ");
    scanf("%d", &NOP);
    y = NOP; // Assign the number of process to variable y

// Use for loop to enter the details of the process like Arrival time and
the Burst Time
for(i=0; i<NOP; i++)
{
printf("\n Enter the Arrival and Burst time of the Process[%d]\n", i+1);
printf(" Arrival time is: \t");  // Accept arrival time
scanf("%d", &at[i]);
printf(" \nBurst time is: \t"); // Accept the Burst time
scanf("%d", &bt[i]);
temp[i] = bt[i]; // store the burst time in temp array
}
// Accept the Time qunat
printf("Enter the Time Quantum for the process: \t");
scanf("%d", &quant);
// Display the process No, burst time, Turn Around Time and the
waiting time
printf("\n Process No \t\t Burst Time \t\t TAT \t\t Waiting Time ");
for(sum=0, i = 0; y!=0; )
{
if(temp[i] <= quant && temp[i] > 0) // define the conditions
{
    sum = sum + temp[i];
    temp[i] = 0;
    count=1;
    }
    else if(temp[i] > 0)
    {
        temp[i] = temp[i] - quant;
        sum = sum + quant;
    }
    if(temp[i]==0 && count==1)
    {
```

```
        y--; //decrement the process no.
        printf("\nProcess No[%d] \t\t %d\t\t\t\t %d\t\t\t %d", i+1, bt[i],
sum-at[i], sum-at[i]-bt[i]);
        wt = wt+sum-at[i]-bt[i];
        tat = tat+sum-at[i];
        count =0;
    }
    if(i==NOP-1)
    {
        i=0;
    }
    else if(at[i+1]<=sum)
    {
        i++;
    }
    else
    {
        i=0;
    }
}
// represents the average waiting time and Turn Around time
avg_wt = wt * 1.0/NOP;
avg_tat = tat * 1.0/NOP;
printf("\n Average Turn Around Time: \t%f", avg_wt);
printf("\n Average Waiting Time: \t%f", avg_tat);
//getch();
return 0;
}
```

Input and Output Section:

```
Total number of process in the system: 4

 Enter the Arrival and Burst time of the Process[1]
 Arrival time is:       0

Burst time is:  8

 Enter the Arrival and Burst time of the Process[2]
 Arrival time is:       1
```

```
Burst time is:  5

 Enter the Arrival and Burst time of the Process[3]
 Arrival time is:       2

Burst time is:  10

 Enter the Arrival and Burst time of the Process[4]
 Arrival time is:       3

Burst time is:  11
Enter the Time Quantum for the process:         6

 Process No          Burst Time          TAT          Waiting Time
Process No[2]         5                   10              5
Process No[1]         8                   25              17
Process No[3]         10                   27              17
Process No[4]         11                   31              20
 Average Turn Around Time:      14.750000
 Average Waiting Time:  23.250000
```

8. Write program to implement SJF scheduling algorithm.

Program:

```
sanjoy@SANJUVAI:~$ gedit sjf.cpp
#include <bits/stdc++.h>
using namespace std;
//structure for every process
struct Process {
   int pid; // Process ID
   int bt; // Burst Time
   int art; // Arrival Time
};
void findTurnAroundTime(Process proc[], int n, int wt[], int tat[]) {
   for (int i = 0; i < n; i++)
   tat[i] = proc[i].bt + wt[i];
}
//waiting time of all process
void findWaitingTime(Process proc[], int n, int wt[]) {
   int rt[n];
```

```
  for (int i = 0; i < n; i++)
  rt[i] = proc[i].bt;
  int complete = 0, t = 0, minm = INT_MAX;
  int shortest = 0, finish_time;
  bool check = false;
  while (complete != n) {
    for (int j = 0; j < n; j++) {
      if ((proc[j].art <= t) && (rt[j] < minm) && rt[j] > 0) {
        minm = rt[j];
        shortest = j;
        check = true;
      }
    }
    if (check == false) {
      t++;
      continue;
    }
    // decrementing the remaining time
    rt[shortest]--;
    minm = rt[shortest];
    if (minm == 0)
      minm = INT_MAX;
      // If a process gets completely
      // executed
      if (rt[shortest] == 0) {
        complete++;
        check = false;
        finish_time = t + 1;
        // Calculate waiting time
        wt[shortest] = finish_time -
        proc[shortest].bt -
        proc[shortest].art;
        if (wt[shortest] < 0)
          wt[shortest] = 0;
      }
      // Increment time
      t++;
  }
}
// Function to calculate average time
```

```
void findavgTime(Process proc[], int n) {
   int wt[n], tat[n], total_wt = 0,
   total_tat = 0;
   // Function to find waiting time of all
   // processes
   findWaitingTime(proc, n, wt);
   // Function to find turn around time for
   // all processes
   findTurnAroundTime(proc, n, wt, tat);
   cout << "Processes " << " Burst time " << " Waiting time " << " Turn
around time\n";
   for (int i = 0; i < n; i++) {
      total_wt = total_wt + wt[i];
      total_tat = total_tat + tat[i];
      cout << " " << proc[i].pid << "\t\t" << proc[i].bt << "\t\t " << wt[i]
<< "\t\t " << tat[i] << endl;
   }
cout << "\nAverage waiting time = " << (float)total_wt / (float)n;
cout << "\nAverage turn around time = " << (float)total_tat / (float)n;
}
int main() {
   Process proc[] = { { 1, 5, 1 }, { 2, 3, 1 }, { 3, 6, 2 }, { 4, 5, 3 } };
   int n = sizeof(proc) / sizeof(proc[0]);
   findavgTime(proc, n);
   return 0;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ g++ sjf.cpp
sanjoy@SANJUVAI:~$ ./a.out
Processes  Burst time  Waiting time  Turn around time
 1          5          3          8
 2          3          0          3
 3          6          12          18
 4          5          6          11

Average waiting time = 5.25
Average turn around time = 10
```

9. Write program to calculate sum of n numbers using thread library.

Program:

```
sanjoy@SANJUVAI:~$ gedit thread.c
#include <pthread.h>
#include <stdlib.h>
#include <stdio.h>
typedef struct data{
   int* arr;
   int thread_num;
} data;
int arrSize = 10;
void* halfSum(void* p){
   data* ptr = (data*)p;
   int n = ptr->thread_num;
   // Declare sum dynamically to return to join:
   int* thread_sum = (int*) calloc(1, sizeof(int));

   if(n == 0){
      for(int i = 0; i < arrSize/2; i++)
         thread_sum[0] = thread_sum[0] + ptr->arr[i];
   }
   else{
      for(int i = arrSize/2; i < arrSize; i++)
         thread_sum[0] = thread_sum[0] + ptr->arr[i];
   }

   pthread_exit(thread_sum);
}
int main(void){
   // Declare integer array [1,2,3,4,5,6,7,8,9,10]:
   int* int_arr = (int*) calloc(arrSize, sizeof(int));
   for(int i = 0; i < arrSize; i++)
      int_arr[i] = i + 1;
   // Declare arguments for both threads:
   data thread_data[2];
   thread_data[0].thread_num = 0;
   thread_data[0].arr = int_arr;
   thread_data[1].thread_num = 1;
   thread_data[1].arr = int_arr;
```

```
    // Declare thread IDs:
    pthread_t tid[2];
    // Start both threads:
    pthread_create(&tid[0], NULL, halfSum, &thread_data[0]);
    pthread_create(&tid[1], NULL, halfSum, &thread_data[1]);
    // Declare space for sum:
    int* sum0;
    int* sum1;
    // Retrieve sum of threads:
    pthread_join(tid[0], (void**)&sum0);
    pthread_join(tid[1], (void**)&sum1);
    printf("Sum of whole array = %i\n", *sum0 + *sum1);
    return 0;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ gcc thread.c -o a.out
sanjoy@SANJUVAI:~$ ./a.out
Sum of whole array = 55
```

10. ***Write a program to implement first-fit, best-fit and worst-fit allocation strategies.***

First-fit allocation strategies:

Program:

```
sanjoy@SANJUVAI:~$ gedit firstfit.c
// C implementation of First - Fit algorithm
#include<stdio.h>

// Function to allocate memory to blocks as per First fit algorithm
void firstFit(int blockSize[], int m, int processSize[], int n)
{
    int i, j;
    // Stores block id of the block allocated to a process
    int allocation[n];

    // Initially no block is assigned to any process
    for(i = 0; i < n; i++)
    {
```

```
            allocation[i] = -1;
        }
        for (i = 0; i < n; i++)        //here, n -> number of processes
        {
            for (j = 0; j < m; j++)        //here, m -> number of blocks
            {
                if (blockSize[j] >= processSize[i])
                {
                    // allocating block j to the ith process
                    allocation[i] = j;

                    // Reduce available memory in this block.
                    blockSize[j] -= processSize[i];

                    break; //go to the next process in the queue
                }
            }
        }

        printf("\nProcess No.\tProcess Size\tBlock no.\n");
        for (int i = 0; i < n; i++)
        {
            printf(" %i\t\t", i+1);
            printf("%i\t\t", processSize[i]);
            if (allocation[i] != -1)
                printf("%i", allocation[i] + 1);
            else
                printf("Not Allocated");
            printf("\n");
        }
    }

    int main()
    {
        int m; //number of blocks in the memory
        int n; //number of processes in the input queue
        int blockSize[] = {100, 500, 200, 300, 600};
        int processSize[] = {212, 417, 112, 426};
        m = sizeof(blockSize) / sizeof(blockSize[0]);
        n = sizeof(processSize) / sizeof(processSize[0]);
```

```
    firstFit(blockSize, m, processSize, n);

    return 0 ;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ gcc firstfit.c -o a.out
sanjoy@SANJUVAI:~$ ./a.out
```

Process No.	Process Size	Block no.
1	212	2
2	417	5
3	112	2
4	426	Not Allocated

Best-fit allocation strategies:

Program:

```
sanjoy@SANJUVAI:~$ touch bestfit.cpp
// C++ implementation of Best - Fit algorithm
#include<iostream>
using namespace std;

// Method to allocate memory to blocks as per Best fit algorithm
void bestFit(int blockSize[], int m, int processSize[], int n)
{
    // Stores block id of the block allocated to a process
    int allocation[n];

    // Initially no block is assigned to any process
    for (int i = 0; i < n; i++)
            allocation[i] = -1;

    // pick each process and find suitable blocks according to its size ad
assign to it
    for (int i = 0; i < n; i++)
    {
            // Find the best fit block for current process
```

```
            int bestIdx = -1;
            for (int j = 0; j < m; j++)
            {
                if (blockSize[j] >= processSize[i])
                {
                    if (bestIdx == -1)
                        bestIdx = j;
                    else if (blockSize[bestIdx] > blockSize[j])
                        bestIdx = j;
                }
            }

            // If we could find a block for current process
            if (bestIdx != -1)
            {
                // allocate block j to p[i] process
                allocation[i] = bestIdx;

                // Reduce available memory in this block.
                blockSize[bestIdx] -= processSize[i];
            }
        }

        cout << "\nProcess No.\tProcess Size\tBlock no.\n";
        for (int i = 0; i < n; i++)
        {
            cout << " " << i+1 << "\t\t" << processSize[i] << "\t\t";
            if (allocation[i] != -1)
                cout << allocation[i] + 1;
            else
                cout << "Not Allocated";
            cout << endl;
        }
    }
    int main()
    {
        int blockSize[] = {100, 500, 200, 300, 600};
        int processSize[] = {212, 417, 112, 426};
        int m = sizeof(blockSize) / sizeof(blockSize[0]);
        int n = sizeof(processSize) / sizeof(processSize[0]);
```

```
    bestFit(blockSize, m, processSize, n);

    return 0 ;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ g++ bestfit.cpp
sanjoy@SANJUVAI:~$ ./a.out
```

Process No.	Process Size	Block no.
1	212	4
2	417	2
3	112	3
4	426	5

Worst-fit allocation strategies:

Program:

```
sanjoy@SANJUVAI:~$ gedit worstfit.cpp
// C++ implementation of worst - Fit algorithm
#include<bits/stdc++.h>
using namespace std;

// Function to allocate memory to blocks as per worst fit
// algorithm
void worstFit(int blockSize[], int m, int processSize[],

    int n)
{
    // Stores block id of the block allocated to a
    // process
    int allocation[n];

    // Initially no block is assigned to any process
    memset(allocation, -1, sizeof(allocation));

    // pick each process and find suitable blocks
    // according to its size ad assign to it
    for (int i=0; i<n; i++)
```

```
    {
        // Find the best fit block for current process
        int wstIdx = -1;
        for (int j=0; j<m; j++)
        {
            if (blockSize[j] >= processSize[i])
            {
                if (wstIdx == -1)
                    wstIdx = j;
                else if (blockSize[wstIdx] < blockSize[j])
                    wstIdx = j;
            }
        }

        // If we could find a block for current process
        if (wstIdx != -1)
        {
            // allocate block j to p[i] process
            allocation[i] = wstIdx;

            // Reduce available memory in this block.
            blockSize[wstIdx] -= processSize[i];
        }
    }

    cout << "\nProcess No.\tProcess Size\tBlock no.\n";
    for (int i = 0; i < n; i++)
    {
        cout << " " << i+1 << "\t\t" << processSize[i] << "\t\t";
        if (allocation[i] != -1)
            cout << allocation[i] + 1;
        else
            cout << "Not Allocated";
        cout << endl;
    }
}

// Driver code
int main()
{
```

```
    int blockSize[] = {100, 500, 200, 300, 600};
    int processSize[] = {212, 417, 112, 426};
    int m = sizeof(blockSize)/sizeof(blockSize[0]);
    int n = sizeof(processSize)/sizeof(processSize[0]);
    worstFit(blockSize, m, processSize, n);
    return 0 ;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ g++ worstfit.cpp
sanjoy@SANJUVAI:~$ ./a.out

Process No.     Process Size    Block no.
 1              212             5
 2              417             2
 3              112             5
 4              426             Not Allocated
```

C7P: COMPUTER NETWORKS LABORATORY MANUAL
(Course: CC-7)

Experiments

1. Simulate Cyclic Redundancy Check (CRC) error detection algorithm for noisy channel.
2. Simulate and implement stop and wait protocol for noisy channel.
3. Simulate and implement go back n sliding window protocol.
4. Simulate and implement selective repeat sliding window protocol.
5. Simulate and implement distance vector routing algorithm
6. Simulate and implement Dijkstra algorithm for shortest path routing.

1. ***Simulate Cyclic Redundancy Check (CRC) error detection algorithm for noisy channel.***

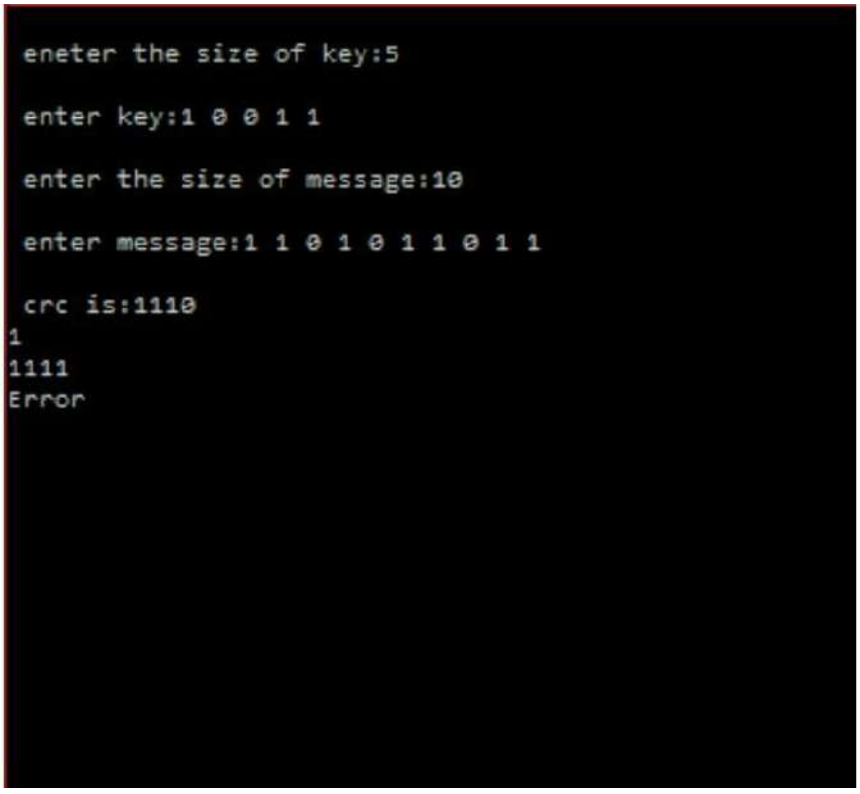
Program:

```
#include<iostream>
#include<conio.h>
#include<stdlib.h>
using namespace std;
void calc(int* temp,int* gen,int size)
{
for(int i=0;i<size;i++)
{
if(temp[i]==gen[i])
temp[i]=0;
else
temp[i]=1;
}
}
int main()
{
int *msg,
*key;
int msize,ksize,i;
cout<<"\n eneter the size of key:";
cin>>ksize;
key=new int[ksize];
cout<<"\n enter key:";
for(i=0;i<ksize;i++)
cin>>key[i];
cout<<"\n enter the size of message:";
cin>>msize;
msg=new int[msize+ksize-1];
cout<<"\n enter message:";
for(i=0;i<msize;i++)
cin>>msg[i];
for(i=msize;i<msize+ksize-1;i++)
msg[i]=0;
int *temp=new int[ksize];
int *zkey=new int[ksize];
for(i=0;i<ksize;i++)
{ temp[i]=msg[i]; zkey[i]=0; }
for(i=ksize-1;i<msize+ksize-1;i++)
```

```
{
temp[ksize-1]=msg[i];
if(temp[0]==0)
calc(temp,zkey,ksize);
else
calc(temp,key,ksize);
for(int j=1;j<ksize;j++)
{ temp[j-1]=temp[j]; }
}
cout<<"\n crc is:";
for(i=0;i<ksize-1;i++)
cout<<temp[i];
for(int i=msize,j=0;i<ksize
+msize-1,j<ksize-1;i++,j++)
{
msg[i]=temp[j];
}
int n=rand()%(ksize+msize+5);
msg[n]=!msg[n];
cout<<"\n"<<n<<"\n";

for(i=0;i<ksize;i++)
{
temp[i]=msg[i]; zkey[i]=0; }

for(i=ksize-1;i<msize+ksize-1;i++)
{
temp[ksize-1]=msg[i];

if(temp[0]==0)

calc(temp,zkey,ksize);

else
calc(temp,key,ksize);

for(int j=1;j<ksize;j++)
{ temp[j-1]=temp[j]; }

}
for(int i=0;i<ksize-1;i++)
{ cout<<temp[i];
```

```
}
int flag=1;
for(int i=0;i<ksize-1;i++)
{ if(temp[i]==1)
flag=0;
}
if(flag==0)
cout<<"\nError";
else
cout<<"\nNo error";
getch();
return 0;
}
```

Input and Output Section:



2. Simulate and implement stop and wait protocol for noisy channel.

Program:

```
#include<iostream>
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<dos.h>
using namespace std;
#define time 5
#define max_seq 1
#define tot_pack 5
int randn(int n)
{
   return rand()%n + 1;
}
typedef struct
{
      int data;
}packet;
typedef struct
{
       int kind;
       int seq;
       int ack;
       packet info;
}frame;
typedef enum{ frame_arrival,error,time_out}event_type;
frame data1;
//creating prototype
void from_network_layer(packet *);
void to_physical_layer(frame *);
void to_network_layer(packet *);
void from_physical_layer(frame*);
void sender();
void receiver();
void wait_for_event_sender(event_type *);
void wait_for_event_receiver(event_type *);
#define inc(k) if(k<max_seq)k++;else k=0;
int i=1;
char turn;
int disc=0;
int main()
```

```
{
  while(!disc)
  {  sender();
    // delay(400);
     receiver();
  }
   getchar();
}
void sender()
{
    static int frame_to_send=0;
    static frame s;
    packet buffer;
    event_type event;
    static int flag=0;       //first place
    if (flag==0)
    {
  from_network_layer(&buffer);
  s.info=buffer;
  s.seq=frame_to_send;
  cout<<"\nsender information \t"<<s.info.data<<"\n";
  cout<<"\nsequence no. \t"<<s.seq;
  turn='r';
  to_physical_layer(&s);
  flag=1;
    }
 wait_for_event_sender(&event);
 if(turn=='s')
 {
    if(event==frame_arrival)
     {
     from_network_layer(&buffer);
     inc(frame_to_send);
     s.info=buffer;
     s.seq=frame_to_send;
     cout<<"\nsender information \t"<<s.info.data<<"\n";
     cout<<"\nsequence no. \t"<<s.seq<<"\n";
     getch();
     turn='r';
     to_physical_layer(&s);
     }
 }
}                  //end of sender function
```

```
void from_network_layer(packet *buffer)
{
    (*buffer).data=i;
    i++;
}                       //end of from network layer function


void to_physical_layer(frame *s)
{


    data1=*s;
}           //end of to physical layer function

void wait_for_event_sender(event_type *e)
{
     static int timer=0;
     if(turn=='s')
     {   timer++;
   //timer=0;
   return ;
  }


  else                      //event is frame arrival
    {
      timer=0;
      *e=frame_arrival;
    }


}            //end of wait for event function


void receiver()
{
    static int frame_expected=0;
    frame s,r;
    event_type event;
    wait_for_event_receiver(&event);
    if(turn=='r')
    {  if(event==frame_arrival)
```

```
    {
        from_physical_layer(&r);
        if(r.seq==frame_expected)
    {
      to_network_layer(&r.info);
      inc (frame_expected);
    }
    else
    cout<<"\nReceiver :Acknowledgement resent \n";
    getch();
    turn='s';
    to_physical_layer(&s);
      }


     }
}                    //end of receiver function



void wait_for_event_receiver(event_type *e)
{
     if(turn=='r')
     {
   *e=frame_arrival;
     }
}


void from_physical_layer(frame *buffer)
{
    *buffer=data1;
}


void to_network_layer(packet *buffer)
{
     cout<<"\nReceiver : packet received \t"<< i-1;
     cout<<"\n Acknowledgement  sent \t";
     getch();
     if(i>tot_pack)
      { disc=1;
```

```
 cout<<"\ndiscontinue\n";
    }
}
```

Input and Output Section:

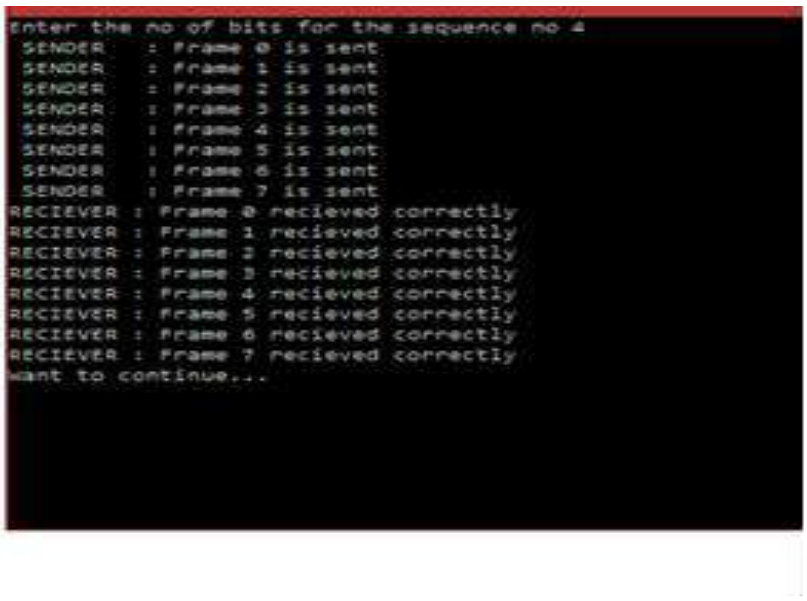
3. Simulate and implement go back n sliding window protocol.

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
char sender[50],receiver[50];
int i,winsize;
printf("\n ENTER THE WINDOWS SIZE : ");
scanf("%d",&winsize);
printf("\n SENDER WINDOW IS EXPANDED TO STORE MESSAGE OR
WINDOW \n");
printf("\n ENTER THE DATA TO BE SENT: ");
fflush(stdin);
gets(sender);
for(i=0;i<winsize;i++)
receiver[i]=sender[i];
receiver[i]=NULL;
printf("\n MESSAGE SEND BY THE SENDER:\n");
puts(sender);
```

```
printf("\n WINDOW SIZE OF RECEIVER IS EXPANDED\n");
 printf("\n ACKNOWLEDGEMENT FROM RECEIVER \n");
for(i=0;i<winsize;i++);
printf("\n ACK:%d",i);
 printf("\n MESSAGE RECEIVED BY RECEIVER IS : ");
 puts(receiver);
 printf("\n WINDOW SIZE OF RECEIVER IS SHRINKED \n");
getch();
}
```

Input and Output Section:



4. Simulate and implement selective repeat sliding window protocol.

Program:

```
#include<iostream>
using namespace std;
#include<conio.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>
#define TOT_FRAMES 500
#define FRAMES_SEND 10
class sel_repeat
{
private:
int fr_send_at_instance;
int arr[TOT_FRAMES];
```

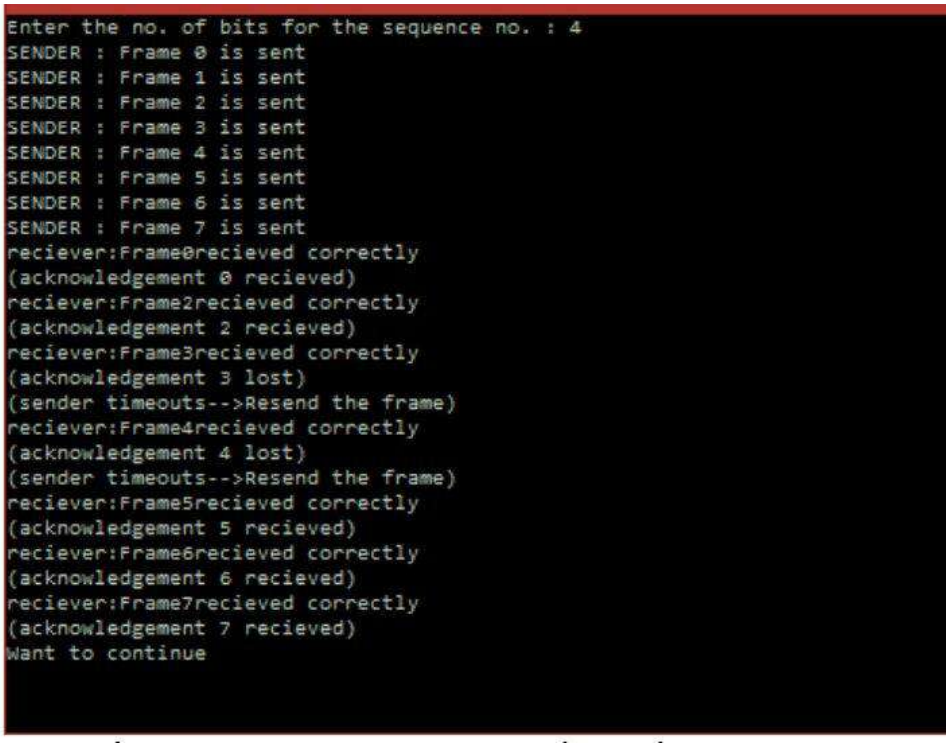
```
int send[FRAMES_SEND];
int rcvd[FRAMES_SEND];
char rcvd_ack[FRAMES_SEND];
int sw;
int rw;
public:
void input();
void sender(int);
void receiver(int);
};
void sel_repeat::input()
{
int n;
int m;
int i;
cout<<"Enter the no. of bits for the sequence no. : ";
cin>>n;
m=pow(2,n);
int t=0;
fr_send_at_instance=(m/2);

for(i=0;i<TOT_FRAMES;i++)
{
arr[i]=t;
t=(t+1)%m;
}
for(i=0;i<fr_send_at_instance;i++)
{
send[i]=arr[i];
rcvd[i]=arr[i];
rcvd_ack[i]='n';
}
rw=sw=fr_send_at_instance;
sender(m);
}
void sel_repeat::sender(int m)
{
for(int i=0;i<fr_send_at_instance;i++)
{
if(rcvd_ack[i]=='n')
cout<<"SENDER : Frame "<<send[i]<<" is sent\n";
}
receiver(m);
```

```
}
void sel_repeat::receiver(int m)
{
time_t t;
int f;
int j;
int f1;
int a1;
char ch;
srand((unsigned)time(&t));
for(int i=0;i<fr_send_at_instance;i++)
{
if(rcvd_ack[i]=='n')
{

f=rand()%10;
if(f!=5)
{
for(int j=0;j<fr_send_at_instance;j++)
if(rcvd[j]==send[i])
{
cout<<"reciever:Frame"<<rcvd[j]<<"recieved correctly\n";
rcvd[j]=arr[rw];
rw=(rw+1)%m;
break;
}
int j;
if(j==fr_send_at_instance)
cout<<"reciever:Duplicate frame"<<send[i]<<"discarded\n";
a1=rand()%5;
if(a1==3)
{
cout<<"(acknowledgement "<<send[i]<<" lost)\n";
cout<<"(sender timeouts-->Resend the frame)\n";
rcvd_ack[i]='n';
}
else
{
cout<<"(acknowledgement "<<send[i]<<" recieved)\n";
rcvd_ack[i]='p';
}
}
else
```

```
{int ld=rand()%2;

if(ld==0)
{
cout<<"RECEIVER : Frame "<<send[i]<<" is damaged\n";
cout<<"RECEIVER : Negative Acknowledgement "<<send[i]<<" sent\n";
}
else
{
cout<<"RECEIVER : Frame "<<send[i]<<" is lost\n";
cout<<"(SENDER TIMEOUTS-->RESEND THE FRAME)\n";
}
rcvd_ack[i]='n';
}
}
}
for(int j=0;j<fr_send_at_instance;j++)
{
if(rcvd_ack[j]=='n')
break;
}
int i=0;
for(int k=j;k<fr_send_at_instance;k++)
{
send[i]=send[k];
if(rcvd_ack[k]=='n')
rcvd_ack[i]='n';
else
rcvd_ack[i]='p';
i++;
}
if(i!=fr_send_at_instance)
{
for(int k=i;k<fr_send_at_instance;k++)
{
send[k]=arr[sw];
sw=(sw+1)%m;
rcvd_ack[k]='n';
}
}
cout<<"Want to continue";
cin>>ch;
cout<<"\n";
```

```
if(ch=='y')
sender(m);
else
exit(0);
}
int main()
{
sel_repeat sr;
sr.input();

 }
```

Input and Output Section:



5. Simulate and implement distance vector routing algorithm.

Program:

```
#include<stdio.h>
#include<iostream>
using namespace std;

struct node
{
   unsigned dist[6];
```

```
    unsigned from[6];
}DVR[10];
int main()
{
    cout<<"\n\n PROGRAM TO IMPLEMENT DISTANCE VECTOR
ROUTING ALGORITHM ";
    int costmat[6][6];
    int nodes, i, j, k;
    cout<<"\n\n Enter the number of nodes : ";
    cin>>nodes; //Enter the nodes
    cout<<"\n Enter the cost matrix : \n" ;
    for(i = 0; i < nodes; i++)
     {
       for(j = 0; j < nodes; j++)
       {
          cin>>costmat[i][j];
          costmat[i][i] = 0;
          DVR[i].dist[j] = costmat[i][j]; //initialise the distance equal to cost
matrix
          DVR[i].from[j] = j;
       }
    }
          for(i = 0; i < nodes; i++) //We choose arbitary vertex k and we
calculate the
          //direct distance from the node i to k using the cost matrix and add
the distance from k to node j
          for(j = i+1; j < nodes; j++)
          for(k = 0; k < nodes; k++)
             if(DVR[i].dist[j] > costmat[i][k] + DVR[k].dist[j])
             {   //We calculate the minimum distance
                DVR[i].dist[j] = DVR[i].dist[k] + DVR[k].dist[j];
                DVR[j].dist[i] = DVR[i].dist[j];
                DVR[i].from[j] = k;
                DVR[j].from[i] = k;

             }
       for(i = 0; i < nodes; i++)
       {
          cout<<"\n\n For router: "<<i+1;
          for(j = 0; j < nodes; j++)
             cout<<"\t\n node "<<j+1<<" via "<<DVR[i].from[j]+1<<"
Distance "<<DVR[i].dist[j];
       }
```

```
    cout<<" \n\n ";
    return 0;
}
```

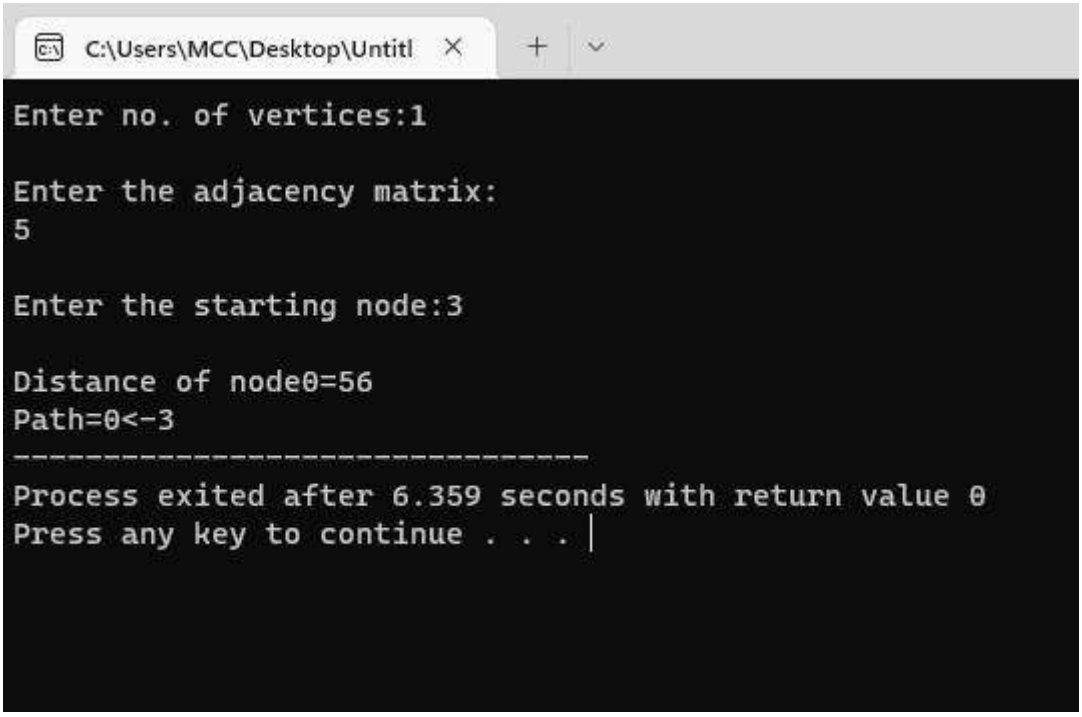
6. Simulate and implement Dijkstra algorithm for shortest path routing.

Program:

```
#include<iostream>
#include<conio.h>
#include<stdio.h>
using namespace std;
int shortest(int ,int);
int cost[10][10],dist[20],i,j,n,k,m,S[20],v,totcost,path[20],p;
main()
{
int c;
cout <<"enter no of vertices";
cin >> n;
cout <<"enter no of edges";
cin >>m;
cout <<"\nenter\nEDGE Cost\n";
for(k=1;k<=m;k++)
{
cin >> i >> j >>c;
cost[i][j]=c;
}
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
if(cost[i][j]==0)
cost[i][j]=31999;
cout <<"enter initial vertex";
cin >>v;
cout << v<<"\n";
shortest(v,n);
 }
int shortest(int v,int n)
{
int min;
for(i=1;i<=n;i++)
{
```

```
S[i]=0;
dist[i]=cost[v][i];
}
path[++p]=v;
S[v]=1;
dist[v]=0;
for(i=2;i<=n-1;i++)
{
k=-1;
min=31999;
for(j=1;j<=n;j++)
{
if(dist[j]<min && S[j]!=1)
{
min=dist[j];
k=j;
}
}

if(cost[v][k]<=dist[k])
p=1;
path[++p]=k;
for(j=1;j<=p;j++)
cout<<path[j];
cout <<"\n";
//cout <<k;
S[k]=1;
for(j=1;j<=n;j++)
if(cost[k][j]!=31999 && dist[j]>=dist[k]+cost[k][j] && S[j]!=1)
 dist[j]=dist[k]+cost[k][j];
}
}
```

Input and Output Section:



INTRODUCTION TO C/C++ PROGRAMMING LABORATORY MANUAL
(Course Name: GE-3)

SL. No.	Experiments
1.	Write a program to find greatest of three numbers.
2.	Write a program to find gross salary of a person
3.	Write a program to find grade of a student given his marks.
4.	Write a program to find divisor or factorial of a given number.
5.	Write a program to print first ten natural numbers.
6.	Write a program to print first ten even and odd numbers.
7.	Write a program to find grade of a list of students given their marks using structure.
8.	Create Matrix class. Write a menu-driven program to perform following Matrix operations (2-D array implementation): a) Sum b) Difference c) Product d) Transpose

1. ***Write a program to find greatest of three numbers.***

Program:

```
#include<stdio.h>
#include<conio.h>
int main()
{
   int a,b,c;
   printf("Enter the three number \n");
   scanf("%d%d%d",&a,&b,&c);
   printf("Three number are: \n a=%d \n b=%d \n c=%d \n",a,b,c);
   if(a>b && a>c){
      printf("The greatest number of three is a=%d",a);
   }
   else if(b>c){
      printf("The greatest number of three is b=%d",b);
   }
   else{
      printf("The greatest number of three is c=%d",c);
   }
   getch();
   return 0;
}
```

Input and Output Section:

```
Enter the three number
10 30 20
Three number are:
 a=10
 b=30
 c=20
The greatest number of three is b=30
Enter the three number
30 59 99
Three number are:
 a=30
 b=59
 c=99
The greatest number of three is c=99
```

2. Write a program to find gross salary of a person.

Write a C program to input basic salary of an employee and calculate gross salary according to given conditions.

(i) If, Basic Salary <= 10000 then, HRA = 20%, DA = 80%

(ii) If, Basic Salary is between 10001 to 20000 then, HRA=25%, DA= 90%

(iii) If, basic salary >= 20001 then, HRA = 30%, da = 95%

Logic to find gross salary of an employee:

Gross salary is the final salary computed after the addition of DA, HRA and other allowances. The formula of DA and HRA is

DA=Basic_Salary*(DA/100)

HRA=Basic_Salary*(HRA/100)

Program:

```
#include <stdio.h>
#include<conio.h>
int main()
{
   float Basic_Salary, Gross_Salary, da, hra;
   printf("Enter basic salary of an employee: ");
   scanf("%f", &Basic_Salary);
   //Calculate D.A and H.R.A according to specified conditions
   if(Basic_Salary<=10000)
   {
      da  = Basic_Salary*(80.0/100.0);
      hra = Basic_Salary*(20.0/100.0);
   }
   else if(Basic_Salary<=20000)
   {
      da  = Basic_Salary*(90.0/100.0);
      hra = Basic_Salary*(25./100.);
   }
   else
   {
      da  = Basic_Salary*(95./100.);
      hra = Basic_Salary*(30./100.);
   }
   // Calculate gross salary
   Gross_Salary = Basic_Salary + hra + da;
```

```
    printf("Gross salary of employee = %.2f", Gross_Salary);
    getch();
    return 0;
}
```

Input and Output Section:

Enter basic salary of an employee: 22000
Gross salary of employee = 49500.00
Enter basic salary of an employee: 9900
Gross salary of employee = 19800.00

3. Write a program to find grade of a student given his marks.

Check the grade of the students based on marks.
First of all, we will take input a mark of subject from the candidate and according to following condition we will calculate the grade.

a. If marks <50 then Grade is F
b. if marks >=50 <60 then Grade is D
c. if marks >=60 <70 then Grade is C
d. if marks >=70 <80 then Grade is B
e. if marks >=80 <90 then Grade is A
f. if marks >=90 then Grade is A+

Program:

```
#include<stdio.h>
#include<conio.h>
int main()
{
   float phy,chem,math,marks;
   printf("Enter your three subject marks  \n");
   scanf("%f%f%f",&phy,&chem,&math);
   marks=(phy+chem+math)/3.0;
   if(marks<0 || marks>100)
   {
      printf("Wrong Marks");
   }
   else if(marks<50)
   {
      printf("%.2f Marks is Grade F",marks);
   }
   else if(marks>=50 && marks<60)
   {
      printf("%.2f Marks is Grade D",marks);
   }
   else if(marks>=60 && marks<70)
   {
      printf("%.2f Marks is Grade C",marks);
   }
   else if(marks>=70 && marks<80)
   {
      printf("%.2f Marks is Grade B",marks);
   }
```

```
    else if(marks>=80 && marks<90)
    {
       printf("%.2f Marks is Grade A",marks);
    }
    else
    {
       printf("%.2f Marks is Grade A+",marks);
    }
    getch();
    return 0;
}
```

Input and Output Section:

```
Enter your three subject marks
99 78 45
74.00 Marks is Grade B
Enter your three subject marks
99 89 100
96.00 Marks is Grade A+
Enter your three subject marks
100 67 199
Wrong Marks
Enter your three subject marks
45 30 40
38.33 Marks is Grade F
```

4. ***Write a program to find divisor or factorial of a given number.***

Factorial of a given number:

Program:

```
#include<stdio.h>
#include<conio.h>
int main()
{
 int i,number;
 long fact=1;
 printf("Enter a number: ");
  scanf("%d",&number);
    for(i=1;i<=number;i++){
      fact=fact*i;
  }
  printf("Factorial of %d is: %ld",number,fact);
  getch();
return 0;
}
```

Input and Output Section:

```
Enter a number: 5
Factorial of 5 is: 120
Enter a number: 9
Factorial of 9 is: 362880
Enter a number: 15
Factorial of 15 is: 1307674368000
```

Divisors of a given number:

Program:

```
#include<stdio.h>
#include<conio.h>
int main()
{
 int i,number;
 printf("Enter a number: ");
  scanf("%d",&number);
  printf("The divisors of %d are: \n",number);
  for (int i=1;i<=number;i++){
```

```
        if (number%i==0)
            printf("%d ",i);
  }
  getch();
return 0;
}
```

Input and Output Section:

```
Enter a number: 100
The divisors of 100 are:
1 2 4 5 10 20 25 50 100
Enter a number: 111
The divisors of 111 are:
1 3 37 111
```

5. Write a program to print first ten natural numbers.

Program:

```
#include<stdio.h>
#include<conio.h>
int main()
{
   int i,number;
   printf("Enter the number: \n");
   scanf("%d",&number);
        printf("The first %d natural numbers are:\n",number);
        for (i=1;i<=number;i++)
        {
                printf("%d ",i);
        }
getch();
return 0;
}
```

Input and Output Section:

```
Enter the number:
10
The first 10 natural numbers are:
1 2 3 4 5 6 7 8 9 10
Enter the number:
15
The first 15 natural numbers are:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
```

6. Write a program to print first ten even and odd numbers.

Program:

```
#include<stdio.h>
#include<conio.h>
int main()
{
        int i;
        printf("10 Odd Number :\t 10 Even Number:\n");
        for(i = 1; i <= 20; i++)
        {
           if((i % 2)!=0)
                {
                        printf("%d\t\t ", i);
                }
                if((i % 2)==0)
                {
                        printf("%d", i);
                        printf("\n");
                }
        }
        getch();
        return 0;
}
```

Input and Output Section:

10 Odd Number:	***10 Even Number:***
1	2
3	4
5	6
7	8
9	10
11	12
13	14
15	16
17	18
19	20

7. ***Write a program to find grade of a list of students given their marks using structure.***

Following the condition

	Student_Percentage	***Grades***
(i)	*>=80*	*A*
(ii)	*>=60*	*B*
(iii)	*>=50*	*C*
(iv)	*>=40*	*D*
(v)	*<40*	*F*

Program:

```
//C program to find students grades in a class through structure
#include<stdio.h>
#include<conio.h>
#include<string.h>
struct stud
{
  char nam[20];
  int phy,chem,math;
  int obtain_mark;
  float per;
  char grad[5];
};
struct stud s[5];
int i;
int main()
{
int std_num;
printf("Enter the number of students: \n");
scanf("%d",&std_num);
 for(i=1; i<=std_num; i++)
 {
  printf("Enter %d student name : ",i);
  scanf("%s",s[i].nam);
  printf("Enter %d student three subject marks: ",i);
  scanf("%d%d%d",&s[i].phy,&s[i].chem,&s[i].math);
  s[i].obtain_mark=s[i].phy+s[i].chem+s[i].math;
  printf("Enter %d student obtained marks = %d \n",i,s[i].obtain_mark);
```

```
 }
 for(i=1; i<=std_num; i++){
  s[i].per=s[i].obtain_mark/3.0;
  //printf("%d student percentage marks: %d \n",i,s[i].per);
 }
 for(i=1; i<=std_num; i++)
 {
  if(s[i].per>=80)
    strcpy(s[i].grad,"A");
  else if(s[i].per>=60)
    strcpy(s[i].grad,"B");
  else if(s[i].per>=50)
    strcpy(s[i].grad,"C");
  else if(s[i].per>=40)
    strcpy(s[i].grad,"D");
  else
    strcpy(s[i].grad,"F");
 }
 for(i=1; i<=std_num; i++)
  printf("\n%d student %s percentage marks %f has obtained grade %s
",i,s[i].nam,s[i].per,s[i].grad);
 getch();
 return 0;
}
```

Input and Output Section:

```
Enter the number of students:
5
Enter 1 student name : Sonali
Enter 1 student three subject marks: 90 67 89
Enter 1 student obtained marks = 246
Enter 2 student name : Saikat
Enter 2 student three subject marks: 65 89 56
Enter 2 student obtained marks = 210
Enter 3 student name : Indrani
Enter 3 student three subject marks: 78 64 45
Enter 3 student obtained marks = 187
Enter 4 student name : Suparna
Enter 4 student three subject marks: 89 65 54
Enter 4 student obtained marks = 208
```

Enter 5 student name : Palash
Enter 5 student three subject marks: 76 86 93
Enter 5 student obtained marks = 255

1 student Sonali percentage marks 82.000000 has obtained grade A
2 student Saikat percentage marks 70.000000 has obtained grade B
3 student Indrani percentage marks 62.333332 has obtained grade B
4 student Suparna percentage marks 69.333336 has obtained grade B
5 student Palash percentage marks 85.000000 has obtained grade A

Enter the number of students:
3
Enter 1 student name : Sanjoy
Enter 1 student three subject marks: 89 98 99
Enter 1 student obtained marks = 286
Enter 2 student name : Subhadeep
Enter 2 student three subject marks: 78 65 66
Enter 2 student obtained marks = 209
Enter 3 student name : Binod
Enter 3 student three subject marks: 23 45 19
Enter 3 student obtained marks = 87

1 student Sanjoy percentage marks 95.333336 has obtained grade A
2 student Subhadeep percentage marks 69.666664 has obtained grade B
3 student Binod percentage marks 29.000000 has obtained grade F

8. Create Matrix class. Write a menu-driven program to perform following Matrix operations (2-D array implementation): a) Sum b) Difference c) Product d) Transpose

Program:

```
#include <iostream>
using namespace std;
class Matrix{
   public:
void AdditionMatrix(int a[10][10], int b[10][10], int r1, int c1, int r2, int
c2){
   int i,j,add[10][10];
            for(i=0;i<r1;i++){
               for(j=0;j<c2;j++){
                        add[i][j]=a[i][j]+b[i][j];
               }
}
cout<<"....Output matrix....\n";
for(i=0;i<r1;i++){
        for(j=0;j<c1;j++){
                cout<<add[i][j]<<" ";
        }
     cout<<endl;
}
         }
void SubtractionMatrix(int a[10][10], int b[10][10], int r1, int c1, int r2,
int c2){
       int i,j,sub[10][10];
            for(i=0;i<r1;i++){
               for(j=0;j<c2;j++){
                        sub[i][j]=a[i][j]-b[i][j];
               }
}
cout<<"....Output matrix....\n";
for(i=0;i<r1;i++){
        for(j=0;j<c1;j++){
                cout<<sub[i][j]<<" ";
        }
     cout<<endl;
      }
```

```
    }
void MultiplicationMatrix(int a[10][10],int b[10][10],int r1,int c1,int
r2,int c2){
    int mul[10][10],i,j,k;
if(c1==r2){
    for(i=0;i<r1;i++){
        for(j=0;j<c2;j++){
                mul[i][j]=0;
                for(k=0;k<c1;k++){
                        mul[i][j]+=a[i][k]*b[k][j];
                }
        }
}
printf("....Output matrix....\n");
for(i=0;i<r1;i++){
        for(j=0;j<c2;j++){
                printf("%d ",mul[i][j]);
        }
     cout<<endl;
    }

}
    else{
       cout<<"can't be multiplied"<<endl;
    }
}
void TransposeMatrix(int a[10][10],int r1,int c1){
    cout<<"Transpose of matrix \n";
    int i,j;
for(i=0;i<c1;i++){
        for(j=0;j<r1;j++){
                printf("%4d",a[j][i]);
        }
        printf("\n");
}
}
};
int main(){
        int a[10][10],b[10][10],r1, c1, r2, c2,ch;
        Matrix m;
```

```
        cout<<"Enter the no. of rows and column of 1st matrix"<<endl;
        cin>>r1>>c1;
        cout<<"Enter the no. of rows and column of 2nd matrix"<<endl;
        cin>>r2>>c2;
        cout<<"Enter the elements of the first matrix"<<endl;
        for(int i=0; i<r1; i++){
                for(int j=0; j<c1; j++){
                        cin>>a[i][j];
                }
        }
        cout<<"Enter the elements of the second matrix"<<endl;
        for(int i=0; i<r2; i++){
                for(int j=0; j<c2; j++){
                        cin>>b[i][j];
                }
        }
        do{
        cout<<"\n1.Addition\n2.Subtraction\n3.Multiplication\n4.Transpos
e\n0.Exit"<<endl;
                cout<<"Enter your choice:";
                cin>>ch;
                switch(ch){
                        case 1: m.AdditionMatrix(a,b,r1,c1,r2,c2);
                           break;
                        case 2: m.SubtractionMatrix(a,b,r1,c1,r2,c2);
                           break;
                        case 3: m.MultiplicationMatrix(a,b,r1,c1,r2,c2);
                           break;
                        case 4: m.TransposeMatrix(a,r1,c1);
                           break;
                        case 0:
                        break;
                        default:
                        cout<<"Wrong choice try again";
                }
        }while(ch!=0);
                return 0;
}
```

Input and Output Section:

```
Enter the no. of rows and column of 1st matrix
3 3
Enter the no. of rows and column of 2nd matrix
3 3
Enter the elements of the first matrix
4 5 6
7 8 9
10 11 12
Enter the elements of the second matrix
1 2 3
7 8 9
4 5 6

1.Addition
2.Subtraction
3.Multiplication
4.Transpose
0.Exit
Enter your choice:1
....Output matrix....
5 7 9
14 16 18
14 16 18

1.Addition
2.Subtraction
3.Multiplication
4.Transpose
0.Exit
Enter your choice:2
....Output matrix....
3 3 3
0 0 0
6 6 6

1.Addition
2.Subtraction
3.Multiplication
4.Transpose
```

```
0.Exit
Enter your choice:3
....Output matrix....
63 78 93
99 123 147
135 168 201

1.Addition
2.Subtraction
3.Multiplication
4.Transpose
0.Exit
Enter your choice:4
Transpose of matrix
  4   7  10
  5   8  11
  6   9  12

1.Addition
2.Subtraction
3.Multiplication
4.Transpose
0.Exit
Enter your choice:0


...Program finished with exit code 0
Press ENTER to exit console.
```

SEC-1P: SOFTWARE LABORATORY BASED ON MATLAB MANUAL

(Course Code: SEC-1P)

Sl. No.	Program Description
1.	WRITE A SCRIPT IN MATLAB TO FIND THE SUM OF NATURALS NUMBERS BETWEEN 1 AND 100
2.	WRITE A SCRIPT IN MATLAB TO FIND THE SUM OF NATURALS NUMBERS BETWEEN TWO SPECIFIED NUMBERS
3.	WRITE A SCRIPT IN MATLAB TO CHECK WHETHER A GIVEN NUMBER IS PRIME OR NOT
4.	WRITE A SCRIPT IN MATLAB TO CHECK WHETHER A GIVEN NUMBER IS DIVISIBLE OR NOT.
5.	WRITE A SCRIPT IN MATLAB TO CHECK WHETHER A GIVEN NUMBER IS PALINDROME OR NOT.
6.	WRITE A SCRIPT IN MATLAB TO GENERATE FIBONACCI SEQUENCE.
7.	WRITE A SCRIPT IN MATLAB TO FIND THE PASCAL TRIANGLE
8.	WRITE A SCRIPT IN MATLAB TO FIND THE PRIME FACTORS OF A POSITIVE INTEGER
9.	WRITE A SCRIPT IN MATLAB TO FIND THE ROOTS OF A QUADRATIC EQUATION
10.	WRITE A SCRIPT IN MATLAB TO FIND THE ADDITION OF TWO MATRICES
11.	WRITE A SCRIPT IN MATLAB TO FIND THE MEAN, MEDIAN, VARIANCE AND STANDARD DEVIATION FOR A SET OF DISCRETE DATA
12.	WRITE A SCRIPT IN MATLAB TO DRAW $sin\ t$ AND $cos\ t$ CURVES IN THE SAME FIGURE WITH DIFFERENT LINE SPECIFICATION
13.	WRITE A SCRIPT IN MATLAB TO DISPLAY THREE-DIMENSIONAL SURFACE PLOT
14.	WRITE A SCRIPT IN MATLAB TO DRAW A PIE CHART FOR A SET OF DATA NUMBER
15.	WRITE A SCRIPT IN MATLAB TO DRAW HISTOGRAPHY FOR A SET OF DATA
16.	WRITE A SCRIPT IN MATLAB TO PLOT 5X^2+15 IN A GIVEN INTERVAL
17.	WRITE A SCRIPT IN MATLAB TO PLOT $Y = \|X\|$ AND $Y = \mathrm{LOG}\,X$ IN A SAME FIGURE.

Programming 1: Write a script in MATLAB to find the sum of naturals numbers between 1 and 100

Code:

```
Sum=0;
for i=1:100
   Sum=Sum+i;
end
fprintf('The sum of natural numbers between 1 and 100 is %d\n', Sum);
```

Input/output:

The sum of natural numbers between 1 and 100 is 5050.

Programming 2: ***Write a script in MATLAB to find the sum of naturals numbers between two specified numbers***

```
m=input('Enter the lower value\n');
n=input('Enter the upper value\n');
Sum=0;
for i=m:n
   Sum=Sum+i;
end
fprintf('The sum of the natural numbers between %d and %d is %d\n',m,n,Sum);
```

Input/output:

Enter the lower value

1

Enter the upper value

50

The sum of the natural numbers between 1 and 50 is 1275.

Programming 3: ***Write a script in MATLAB to check whether a given number is prime or not***

Code:

```
n=input('Enter the number\n');
if isprime(n)==1
    fprintf('The number %d is prime\n',n);
else
    fprintf('The number %d is not prime\n',n);
end
```

Input/output:

Enter the number

25

The number 25 is not prime.

Programming 4: ***Write a script in MATLAB to check whether a given number is divisible or not***

Code:

```
m=input('Enter the dividend\n');
n=input('Enter the divisor\n');
if rem(m,n)==0
    fprintf('The number %d is divisible by %d \n',m,n);
else
    fprintf('The number %d is not divisible by %d \n',m,n);
end
```

Input/output:

Enter the dividend

25

Enter the divisor

5

The number 25 is divisible by 5.

Programming 5: ***Write a script in MATLAB to check whether a given number is palindrome or not***

Code:

```
n=input('Enter the number \n');
P=n;
rev=0;
while(n>0)
    dig=rem(n,10);
    rev=(rev*10)+dig;
    n=floor(n/10);
end
```

Input/output:

Enter the number

111

The number 111 is palindrome.

Programming 6: ***Write a script in MATLAB to generate Fibonacci sequence***

Code:

```
n=input('How many numbers are required\n');
if(n==1)
   fprintf('The first fibonacci number is o\n');
else
   f1=0;
   f2=1;
   fprintf('The first %d fibonacci numbers are \t%d\t%d',n,f1,f2);
   for i=1:n-2
      f=f1+f2;
      f1=f2;
      f2=f;
      fprintf('\t%d',f);
   end
end
fprintf('\n');
```

Input/output:

How many numbers are required

10

The first 10 fibonacci numbers are 0 1 1 2 3 5 8 13 21 34

Programming 7: ***Write a script in MATLAB to find the Pascal triangle***

Code:

```
row=input('Enter the number of rows\n');
b=1
fprintf('The Pascal triangle with %d rows is \n',row);
for n=0:(row-1)
   for s=(25-3*n):-1:1
      fprintf(' ');
   end
   for r=0:n
      if n==0||r==0
         b=1;
      else
         b=b*(n-r+1);
      end
      fprintf('% 6d',b);
   end
   fprintf('\n');
end
```

Input/output:

Enter the number of rows

5

b =

1

The Pascal triangle with 5 rows is

```
          1
        1   1
      1   2   2
    1   3   6   6
  1   4  12  24  24
```

Programming 8: ***Write a script in MATLAB to find the prime factors of a positive integer***

Code:

```
n=input('Enter a positive integer:\n');
fprintf('The prime factors of %d are:\n',n);
for j=2:n
   if rem(n,j)==0
      if isprime(j)==1
         fprintf('\t %d',j);
      end
   end
end
fprintf('\n');
```

Input/output:

Enter a positive integer:

45

The prime factors of 45 are:

3 5

Programming 9: ***Write a script in MATLAB to find the roots of a quadratic equation***

Code:

```
fprintf('Enter the values of the coefficient a, b, c of x^2, x^1 and x^0 respectively:\n');
 a=input('');
 b=input('');
 c=input('');
 if a==0
    fprintf('Not a quadratic equation');
 else
    d=(b*b-4*a*c);
    p=-b/(2*a);
    q=sqrt(abs(d))/(2*a);
    if d >=0
       fprintf('The real roots are %f and %f \n',p+q,p-q);
    else
       fprintf('The complex rots are %f+%fi and %f-%fi \n',p,q,p,q);
    end
 end
```

Input/output:

Enter the values of the coefficient a, b, c of x^2, x^1 and x^0 respectively:

2

6

1

The real roots are -0.177124 and -2.822876

Programming 10: ***Write a script in MATLAB to find the addition of two matrices***

Code:

```
x=input('Enter the first matrix:\n');
y=input('Enter the second matrix:\n');
a=size(x);
b=size(y);
d=a-b;
if any(d)
 fprintf('Adittion is not possible. Matrix order must be same');
else
 z=x+y;
  fprintf ('The required sum of the matrices is:\n');
  disp(z);
end
```

Input/output:

Enter the first matrix:

[2 3; 1 2]

Enter the second matrix:

[4 5; 6 7]

The required sum of the matrices is:

```
    6     8
    7     9
```

Programming 11: ***Write a script in MATLAB to find the mean, median, variance and standard deviation for a set of discrete data***

Code:

```
n=input('Enter the size of the sample:\n');
fprintf('Enter all the sample values:\n');
sum=0;
sumsq=0;
for i=1:n
   x(i)=input(' ');
   sum=sum+x(i);
   sumsq=sumsq+(x(i)*x(i));
end
xbar=sum/n;
var=(sumsq/n)-(xbar*xbar);
fprintf('Mean=%f\n Variance=%f\n Standard deviation =%f\n',xbar,var,sqrt(var));
for i=1:n-1
   for j=i:n
      if x(i)>x(j)
         t=x(i);
         x(i)=x(j);
      end
   end
end
if rem(n,2)==0
   m=(x(n/2)+x(n/2+1))/2;
else
   m=x((n+1)/2);
end
fprintf('Median=%f\n',m);
```

Input/output:

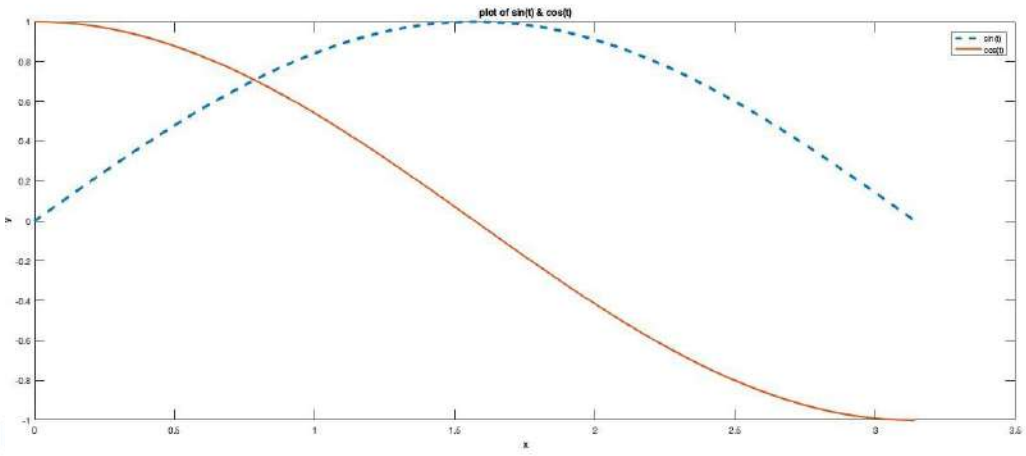
```
Enter the size of the sample:
3
Enter all the sample values:
 1
 2
 3
Mean=2.000000
Variance=0.666667
Standard deviation =0.816497
Median=2.000000
```

Programming 12: *Write a script in MATLAB to draw* sin *t and* cos *t curves in the same figure with different line specification*

Code:

```
t=0:0.001:2*pi;
y=sin(t);
plot(t,y);
hold on
z=cos(t);
plot(t,z);
legend ('sin(t)','cos(t)');
```

Input/output:



Programming 13: ***Write a script in MATLAB to display three-dimensional surface plot***

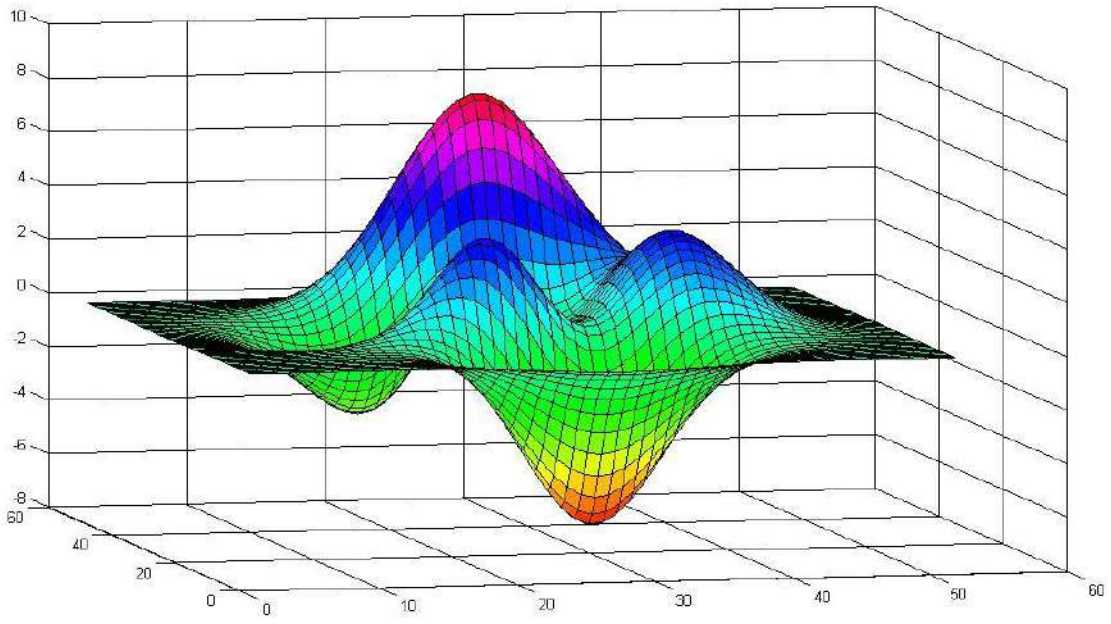
Code:

```
a=input('Enter the value of a:\n');
z=peaks(a);
surface(z);
colormap(hsv);
```

Input/output:

Enter the value of a:

45



Programming 14: ***Write a script in MATLAB to draw a pie chart for a set of data number***

Code:

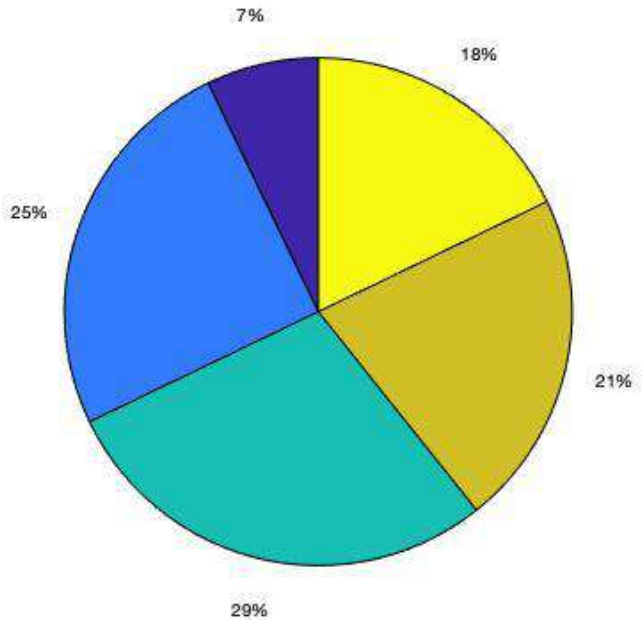
```
X=input('Enter the data:\n');
fprintf('The required pie chart is:\n')
pie(X);
```

Input/output:

Enter the data:

[2 7 8 6 5]

The required pie chart is:



Programming 15: ***Write a script in MATLAB to draw histography for a set of data***

Code:

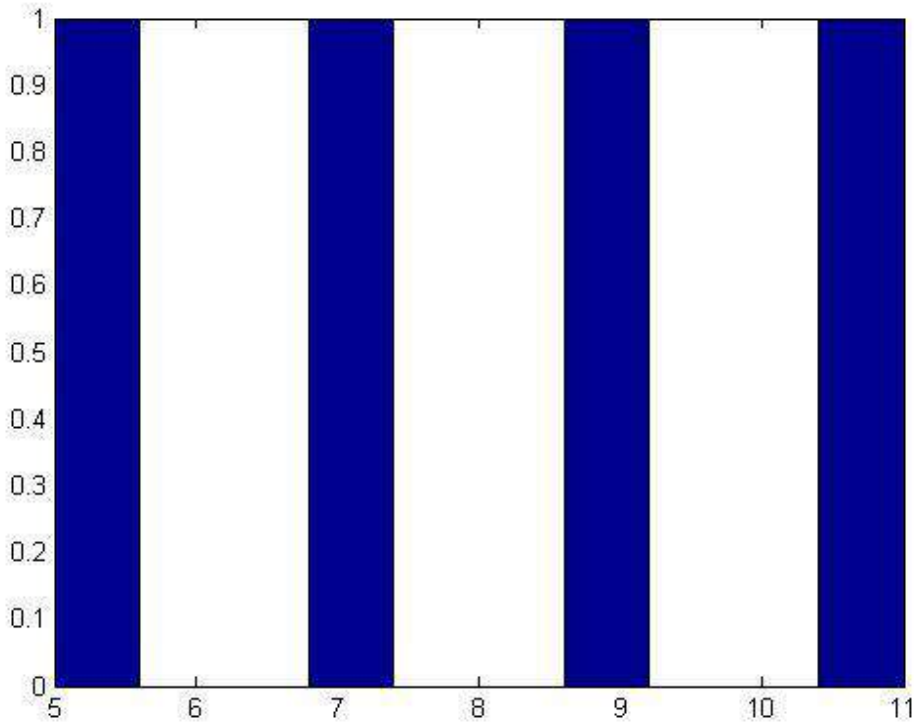
```
X=input('Enter the data:\n')

fprintf('The required histograph is:\n');

hist(X);
```

Input/output:

Enter the data:

[2 7 8 6 5]

The required histograph is:



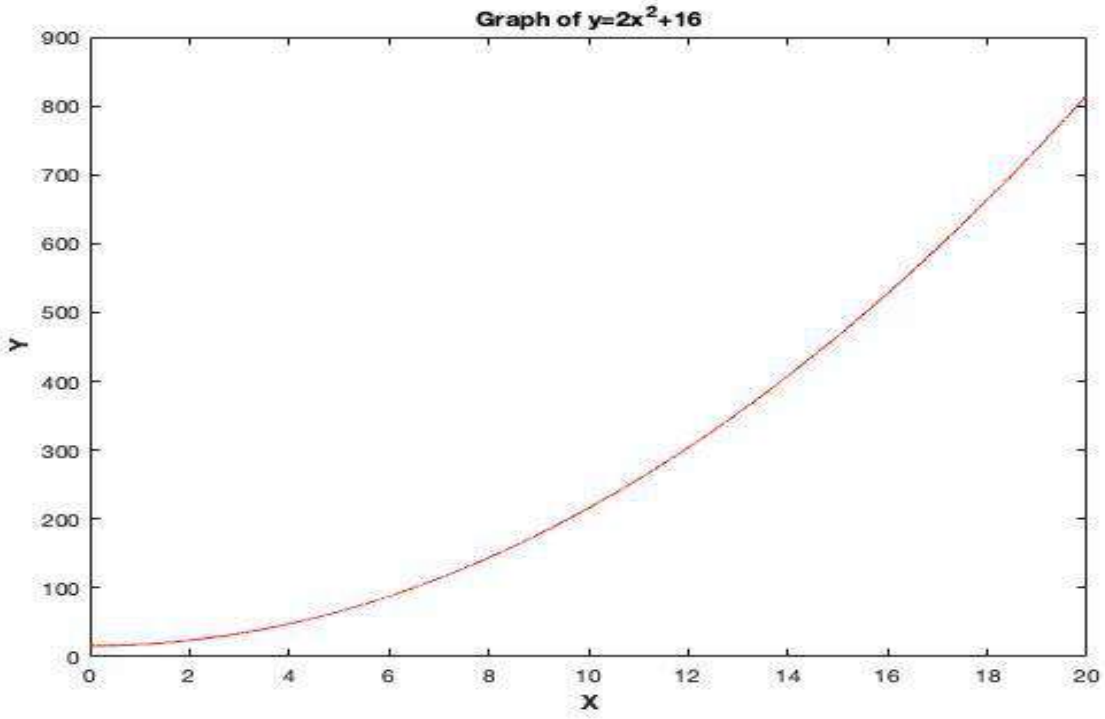
Programming 16: ***Write a script in MATLAB to plot $5x^2 + 15$ in a given interval***

Code:

```
a=input('Enter the lower limit:\n');
b=input('Enter the upper limit:\n');
x=a:0.001:b;
y=2*x.^2+16;
plot(x,y,'r');
title('Graph of y=2x^2+16');
xlabel('\bf X');
ylabel('\bf Y');
```

Input/output:

Enter the lower limit:
0
Enter the upper limit:
20



Programming 17: ***Write a script in MATLAB to plot $y = |x|$ and $y = \log x$ in a same figure***

Code:

```
a=input('Enter the lower limit:\n');
b=input('Enter the upper limit:\n');
x=a:0.001:b;
y=abs(x);
plot(x,y);
hold on
z=log(x);
plot(x,z);
title('\bf plot of abs(x) & log(x)')
% adds title to the plot
xlabel('\bf X');
ylabel('\bf Y');
legend ('|x|','log(x)');
```

Input/output:

Enter the lower limit:
-10
Enter the upper limit:
10