# M.Sc. COMPUTER SCIENCE LAB MANUAL

1st Semester


Prepared By

**Pure and Applied Science Dept.**

Computer Science

## MIDNAPORE CITY COLLEGE

# DATA STRUCTURE LAB MANUAL

## Code: COS-191

# Objectives

- To implement linear and non-linear data structures
- To implement non-linear data structures
- To understand the different operations of search trees
- To implement graph traversal algorithms
- To get familiarized to sorting and searching algorithms
- To implement sorting, searching algorithms.

# INDEX

# Programs

1. Write a C Program to perform following operations on Singly Linked List ADT :
   i. Create
   ii. Insert
   iii. Delete
   iv. Display
   v. Exit

PROGRAM:

```
#include<stdio.h>
#include<string.h>
#include<alloc.h>
#include<conio.h>

struct student
{
 int sno;
 char sname[25];
 int m1,m2,m3;
 int tot;
 float per;
 struct student *next;
 };

 void main()
 {
 char cont='y',choice;
 char sear[20];
 int ch,i,nsno;
 struct student *pp,*fp,*p,*p1,*np;
 clrscr();

 do
 {
 printf("\n 1.Creation");
 printf("\n 2.Traversing the Linked List");
 printf("\n 3.Locating the particular element");
 printf("\n 4.Inssertion");
 printf("\n 5.Deletion");
 printf("\n 6.Quit");
 printf("\n Enter your choice:");
 scanf("%d",&ch);

 switch(ch)
```

```
while(cont=='y')
        {
        printf("enter student no");
        scanf("%d", &p->sno);
        printf("enter student name");
        scanf("%s", &p->sname);
        printf("enter student mark1");
        scanf("%d", &p->m1);
        printf("enter student mark2");
        scanf("%d", &p->m2);
        printf("enter student mark3");
        scanf("%d", &p->m3);
        (*p).tot=(*p).m1+(*p).m2+(*p).m3;
        (*p).per=(float)(*p).tot/3.0;
        printf("do you want to continue");
        fflush(stdin);
        cont=getchar();
        if(cont=='y')
        p->next=(struct student *)malloc(sizeof(struct student));
        else
        p->next=(struct student *)0;
        p=p->next;
        }
        break;

        case 2:p1=fp;
           printf("traversing the linked list");
           while(p1!=(struct student *)0)
           {
           printf("student no:%d\n",p1->sno);
           printf("student name:%s\n",p1->sname);
           printf("marks1:%d\n",p1->m1);
           printf("marks2:%d\n",p1->m2);
           printf("marks3:%d\n",p1->m3);
           printf("total:%d\n",p1->tot);
           printf("average:%f\n",p1->per);
           getch();
           p1=p1->next;
           }
           break;

        case 3:p1=fp;
           printf("enter name of the student that you want to find out:");
           scanf("%s",sear);
           while(p1!=(struct student *)0)
           {
           i=strcmp(p1->sname,sear);
           if(i==0)
           {
           printf("student no:%d\n",p1->sno);
           printf("student name:%s\n",p1->sname);
           printf("marks1:%d\n",p1->m1);
```

```
printf("marks2:%d\n",p1->m2);
printf("marks3:%d\n",p1->m3);
printf("total:%d\n",p1->tot);
printf("average:%f\n",p1->per);
break;
}
p1=p1->next;
}
if(p1==0&&i!=0)
printf("\n name not found");
break;

case 4:printf("a)at the beginning...\n");
        printf("b)at the middle...\n");
        printf("c)at the end...\n");
        printf("enter your choice");
        scanf("%*c%c",&choice);
        switch(choice)
        {
        case'a':p=(struct student *)malloc(sizeof(struct student));
        printf("enter student number");
        scanf("%d",&p->sno);
        printf("enter student name :");
        scanf("%s",&p->sname);
        printf("enter student mark1:");
        scanf("%d",&p->m1);
        printf("enter student mark2:");
        scanf("%d",&p->m2);
        printf("enter student mark3:");
        scanf("%d*c",&p->m3);
        (*p).tot=(*p).m1+(*p).m2+(*p).m3;
        (*p).per=(float)(*p).tot/3.0;
        p->next=fp;
        fp=p;
        break;
        case'b':p1=fp;
        printf("after which student do you want to insert");
        scanf("%d", &nsno);
        while(p1->sno!=nsno)
        p1=p1->next;
        np=p1->next;
        p=(struct student *)malloc(sizeof(struct student));
        p1->next=p;
        printf("enter student number");
        scanf("%d",&p->sno);
        printf("enter student name :");
        scanf("%s",&p->sname);
        printf("enter student mark1:");
        scanf("%d",&p->m1);
        printf("enter student mark2:");
        scanf("%d",&p->m2);
        printf("enter student mark3:");
        scanf("%d*c",&p->m3);
```

```
        (*p).tot=(*p).m1+(*p).m2+(*p).m3;
        (*p).per=(float)(*p).tot/3.0;
        p->next=np;
        break;
        case'c':p1=fp;
        while(p1->next!=(struct student *)0)
        p1=p1->next;
        p=(struct student *)malloc(sizeof(struct student));
        p1->next=p;
        printf("enter student number");
        scanf("%d",&p->sno);
        printf("enter student name:");
        scanf("%s",&p->sname);
        printf("enter student mark1:");
        scanf("%d",&p->m1);
        printf("enter student mark2:");
        scanf("%d",&p->m2);
        printf("enter student mark3:");
        scanf("%d*c",&p->m3);
        (*p).tot=(*p).m1+(*p).m2+(*p).m3;
        (*p).per=(float)(*p).tot/3.0;
        p->next=(struct student *)0;
        break;
    }
    break;
case 5: printf("which student record u want to delete");
        scanf("%d", &nsno);
        if(fp==NULL)
        {
        printf("the singly linked list is empty");
        break;
        }
        p1=fp;
        if(fp->sno==nsno)
        {
        fp=fp->next;
        p1=fp;
        break;
        }
        else
        {
        p1=fp;
        while(p1->sno!=nsno)
        {
        pp=p1;
```

```
p1=p1->next;
}
pp->next=p1->next;
printf("record deleted");
break;
}
p1=fp;

while(p1->next!=(struct student *)0)
p1=p1->next;
if(p1->next==((struct student *)0)&&(p1->sno==nsno))
{
p1=p1->next-1;
p1->next=((struct student *)0);
printf("no:%d",p1->sno);
break;
}
else
{
printf("the given sno is not found \n");
break;
}
}
} while(ch!=6);
}
```

2. Write a C Program to perform following operations on Doubly Linked List ADT :
   i. Create
   ii. Insert
   iii. Delete
   iv. Display
   v. Exit

PROGRAM:

```
#include <stdio.h>
#include <malloc.h>
#include<process.h>

typedef struct DList_tag
{
        int data;
        struct DList_tag *rlink, *llink;
}node;


/**************Function Declaration Begin**********/
node *DLcreation(node **);
void DLinsertion(node **, node **, int, int);
void DLdeletion(node **, node**);
void DLdisplay(node *, node *);
/**************Function Declaration End**********/

void main()
{
        node *left=NULL,*right;
        int item,pos,ch;

        printf("\n\t\tProgram for doubly linked list\n");
        do
        {
                printf("\n\t\tMenu");
                printf("\n\t\t1.Create");
                printf("\n\t\t2.Insert");
                printf("\n\t\t3.Delete");
                printf("\n\t\t4.Display");
                printf("\n\t\t5.Exit");
                printf("\n\t\tEnter choice : ");
                scanf("%d",&ch);

                switch(ch)
                {
                        case 1:
                                left = DLcreation(&right);
                                break;
                        case 2:
```

```
                                printf("\nEnter data :");
                                scanf("%d",&item);
                                do
                                {
                                        printf("\nEnter position of insertion :");
                                        scanf("%d",&pos);
                                }while(pos < 1);
                                DLinsertion(&left,&right,item,pos);
                                break;
                        case 3:
                                DLdeletion(&left,&right);
                                break;
                        case 4:
                                printf("\n\t**** Doubly linked list *****\n");
                                DLdisplay(left,right);
                                break;
                        case 5:
                                exit(0);
                        default:
                                printf("\n Wrong Choice");
                }
        }while(ch!=5);
        printf("\n");
}

/********** Creating of double linked list MENU **********/
/********** Function Definition begins **********/
node *DLcreation( node **right )
{
        node *left, *new_node;
        int item,ch;
        *right = left = NULL;
        do
        {
                printf("\n\t\tMenu");
                printf("\n\t\t1.Add node");
                printf("\n\t\t2.Quit");
                printf("\n\t\tEnter choice : ");
                scanf("%d",&ch);

                switch(ch)
                {
                        case 1:
                                printf("\n Enter data:");
                                scanf("%d",&item);
                                new_node = (node *)malloc(sizeof(node));
                                new_node->data = item;
                                new_node->rlink = NULL;
                                if(left == NULL)
                                        {
                                                new_node->llink = NULL;
```

```
                                        left = new_node;
                                }
                        else
                                {
                                        new_node->llink = (*right);
                                        (*right)->rlink = new_node;
                                }

                        (*right) = new_node;
                        if(left != NULL)
                                (*right) = new_node;
                        break;
                case 2:
                        break;
                default:
                        printf("\n Wrong Choice");
                }

        }while(ch!=2);
        return left;
}

/********** Function Definition ends **********/


/********** Insertion of node in double linked list **********/
/********** Function Definition begins **********/
void DLinsertion(node **start, node **right,int item, int pos)
{
        node *new_node, *temp;
        int i;
        if((pos == 1) ||((*start) == NULL))
        {
                new_node = (node *)malloc(sizeof(node));
                new_node->data = item;
                new_node->rlink = *start;
                new_node->llink = NULL;
                if((*start) != NULL)
                        (*start)->llink = new_node;
                else
                        (*right) = new_node;
                *start = new_node;
        }
        else
        {
                temp = *start;
                i = 2;
                while((i < pos) && (temp->rlink != NULL))
                {
                        temp = temp->rlink;
                        ++i;
                }
```

```
                new_node = (node *)malloc(sizeof( node));
                new_node->data = item;
                new_node->rlink = temp->rlink;
                if(temp->rlink != NULL)
                        temp->rlink->llink = new_node;
                        new_node->llink = temp;
                        temp->rlink = new_node;
        }
        if(new_node->rlink == NULL)
                *right = new_node;
}
/********** Function Definition ends **********/


/********** Deletion of node in linked list **********/
/********** Function Definition begins **********/
void DLdeletion( node **start, node **right)
{
        node *temp, *prec;
        int item;

        printf("\nElement to be deleted :");
        scanf("%d",&item);

        if(*start != NULL)
        {
                if((*start)->data == item)
                {
                        if((*start)->rlink == NULL)
                                *start = *right = NULL;
                        else
                        {
                                *start = (*start)->rlink;
                                (*start)->llink = NULL;
                        }
                }
                else
                {

                        temp = *start;
                        prec = NULL;
                        while((temp->rlink != NULL) && (temp->data != item))
                        {
                                prec = temp;
                                temp = temp->rlink;
                        }
                        if(temp->data != item)
                                printf("\n Data in the list not found\n");
                        else
                        {
                                if(temp == *right)
```

```
                                        *right = prec;
                                else
                                        temp->rlink->llink = temp->llink;
                                prec->rlink = temp->rlink;
                        }
                }
        }
        else

                printf("\n!!! Empty list !!!\n");
        return;
}

/********** Function Definition ends **********/



/********** Displaying nodes of double linked list **********/
/********** Function Definition begins **********/
void DLdisplay(node *start, node *right)
{
        printf("\n***** Traverse in Forward direction *****\n left->");
        while(start != NULL)
        {
                printf("%d-> ",start->data);
                start = start->rlink;
        }
        printf("right");
        printf("\n***** Traverse in Backward direction *****\n right->");
        while(right != NULL)
        {
                printf("%d-> ",right->data);
                right = right->llink;
        }
        printf("left");
}
/********** Function Definition ends **********/
```

3. Write a C Program to perform following operations on Circularly Linked List ADT :
   i. Create
   ii. Insert
   iii. Delete
   iv. Display
   v. Exit

PROGRAM:

```
#include <stdio.h>
#include <malloc.h>
#include<process.h>

typedef struct Clist_tag
{
        int data;
        struct Clist_tag *link;
}node;



/**************Function Declaration Begin**********/
node *CLcreation(node **);
void CLinsertion(node **,node **,int , int );
int CLdeletion(node **,node **,int );
void CLdisplay(node *, node *);
/**************Function Declaration End**********/

void main()
{
        node *START=NULL,*last;
        int item,pos,ch;

        printf("\n\t\t Program for single circularly linked list\n");
        do
        {
                printf("\n\t\tMenu");
                printf("\n\t\t1. Create");
                printf("\n\t\t2. Insert");
                printf("\n\t\t3. Delete");
                printf("\n\t\t4. Display");
                printf("\n\t\t5. Exit");
                printf("\n\t\tEnter choice : ");
                scanf("%d",&ch);

                switch(ch)
                {
                        case 1 :
                                START = CLcreation(&last);
                                break;
                        case 2:
```

```
                        printf("\nEnter the element to be inserted :");
                        scanf("%d",&item);
                        do
                        {
                          printf("\nEnter the position of insertion :");
                          scanf("%d",&pos);
                        }while(pos < 1);
                        CLinsertion(&START,&last,item,pos);
                 break;
                 case 3:
                        do
                        {
                          printf("\nEnter the position of deletion :");
                          scanf("%d",&pos);
                        }while(pos < 1);
                        if (!CLdeletion(&START,&last,pos))
                        printf("Cannot delete element at position %d",pos);
                        break;
                 case 4:
                        printf("\n\t***** Single Circular linked list *****\n");
                        CLdisplay(START,last);
                        break;
                 case 5:
                        exit(0);
                 default:
                        printf("\n Wrong Choice:");
            }
        }while (ch!=5);
        printf("\n");
}

/********** Creating of circular linked list MENU **********/
/********** Function Definition begins **********/
node *CLcreation(node **last)
{
        node *START, *new_node;
        int temp,ch;
        *last = START = NULL;
        do
        {
        printf("\n\t\t Menu:");
        printf("\n\t\t1.Add node:");
        printf("\n\t\t2.Quit:");
        printf("\n Enter Choice:");
        scanf("%d",&ch);
        switch(ch)
                {
                        case 1:
                        printf("\n Enter data;");
                        scanf("%d",&temp);
```

```
                                new_node = (node *)malloc(sizeof(node));
                                new_node->data = temp;
                                if(START == NULL)
                                        START = new_node;
                                else
                                        (*last)->link = new_node;
                                (*last) = new_node;
                                if(START != NULL)
                                        new_node->link = START;
                                break;
                        case 2:
                                break;
                        default:
                                printf("\n Wrong Choice:");
                }
        }while(ch!=2);

        return START;
}
/********** Function Definition ends **********/

/********** Insertion of node in circular linked list **********/
/********** Function Definition begins **********/
void CLinsertion(node **start,node **last,int item, int pos)
{
        node *new_node, *temp;
        int i;
        new_node = (node *)malloc(sizeof( node));
        new_node->data = item;
        if((pos == 1) ||((*start) == NULL))
        {
                new_node->link = *start;
                *start = new_node;
                if((*last) != NULL)
                        (*last)->link = *start;
                else
                        *last = *start;
        }
        else
        {

                temp = *start;
                i = 2;
                while((i < pos) && (temp->link != (*start)))
                {
                        temp = temp->link;
                        ++i;
                }
                if(temp->link == (*start))
                        *last = new_node;
                        new_node->link = temp->link;
                        temp->link = new_node;
```

```
        }
}
/********** Function Definition ends **********/


/********** Deletion of node in circular linked list **********/
/********** Function Definition begins **********/
int CLdeletion(node **start,node **last,int pos)
{
        node *temp;
        int i,flag = 1;

        if(*start != NULL)
                if(pos == 1)
                {
                        if((*start)->link != *start)
                        {
                                *start = (*start)->link;
                                (*last)->link = *start;
                        }
                        else
                                *start = *last = NULL;
                }
                else
                {

                        temp = *start;
                        i = 2;
                        while((temp->link != (*start)) && (i<pos))
                        {
                                temp = temp->link;
                                ++i;
                        }
                        if(temp->link != *start)
                        {
                                if(temp->link == *last)
                                        *last = temp;
                                        temp->link = temp->link->link;
                        }
                        else
                                flag = 0;
        }
        else
                flag = 0;

        return flag;
}
/********** Function Definition ends **********/

/********** Displaying nodes of circular linked list **********/
/********** Function Definition begins **********/
void CLdisplay(node *start, node *last)
```

```
{
        printf("\nSTART->");
        if(start != NULL)
        {
                do
                {
                        printf("%d-> ",start->data);
                        start = start->link;
                }while(last->link != start);
                printf("START\n");
        }
        else
                printf("NULL\n");
}

/********** Function Definition ends **********/
```

4. Write a C program to add two Polynomials.

```
/* Program to add two polynomials.  */

#include<stdio.h>
void main()
{
  int  c1[10],e1[10],c2[10],e2[10],i,rc[20],re[20],n,m,k,l,j;
  clrscr();
  printf("Enter the highest index of 1st Polynomial : ");
  scanf("%d",&n);
  for(i=n;i>=0;i--)
  {
    printf("Enter the coefficient of x^%d : ",i);
    scanf("%d",&c1[i]); e1[i]=i;
  }
  printf("Enter the highest index of 2nd Polynomial : ");
  scanf("%d",&m);
  for(i=m;i>=0;i--)
  {
    printf("Enter the coefficient of x^%d : ",i);
    scanf("%d",&c2[i]); e2[i]=i;
  }
  printf("\nThe first Polynomial is : \n");
  for(i=n;i>=0;i--)
  {
        printf("%d x^%d",c1[i],e1[i]);
        if(i>0)  printf(" + ");
  }
  printf("\nThe second Polynomial is : \n");
  for(i=m;i>=0;i--)
  {
        printf("%d x^%d",c2[i],e2[i]);
        if(i>0) printf(" + ");
  }
  k=n; l=m; j=0;
  while(k>=0 && l>=0)
  {
    if(k>=0 || l>=0)
    {
        if(e1[k]==e2[l])
        {
         rc[j]=c1[k]+c2[l]; re[j]=e1[k];
         j=j+1; k=k-1; l=l-1;
        }
        else if(e1[k]>e2[l])
        {
         rc[j]=c1[k]; re[j]=e1[k]; j=j+1; k=k-1;
        }
        else if(e1[k]<e2[l])
```

```
            {
              rc[j]=c2[l]; re[j]=e2[l]; j=j+1; l=l-1;
            }
        }
        else if (k==0 && l>0)
        {
            rc[j]=c2[l]; re[j]=e2[l]; j=j+1; l=l-1;
        }
        else if(k>0 && l==0)
        {
            rc[j]=c1[k]; re[j]=e1[k]; j=j+1; k=k-1;
        }
    }
    printf("\nThe Sum of the two Polynomials is : \n");
    j=j-1;
    for(i=0;i<=j;i++)
    {
      printf("%d x^%d",rc[i],re[i]);
      if(i<j) printf(" + ");
    }
    getch();
}
```

5. A) Program to demonstrate the operation of Stacks using array implementation.

```
/* PROGRAM TO EXPLAIN THE WORKING OF A STACK USING ARRAY
IMPLEMENTATION */

#include<stdio.h>
#include<conio.h>
void main()
{
int j,stack[5]={0};
int p=0;
clrscr();
printf("stack of five elements \n");
printf("put zero to exit \n");
while(1)
{
printf(" enter elements: ");
scanf("%d", &stack[p]);
if(stack[p]!=0)
printf("element %d is %d on top of the stack \n\n ",p+1,stack[p]);
else
{
printf(" \n By choice terminated : ");
printf(" \n The stack is filled with %d elements", p);
break;
}
p++;
if(p>4)
{
printf(" the stack is full");
break;
}
}
printf("\n elements of stack are: ");
for (j=0;j<p;j++)
printf("%d",stack[j]);
}
```

b) Program to demonstrate the PUSH and POP operations in stacks.

```
/* PROGRAM TO DEMONSTRATE PUSH AND POP OPERATIONS */

#include<stdio.h>
#include<conio.h>

static int stack[10],top=-1;

void main()
{
void push(int);
void pop(void);
void show(void);
clrscr();

printf("\n\n push operation: ");

push(5);
show();
push(8);
show();
push(9);
show();

printf("\n\n pop operation: ");

show();
pop();
show();
pop();
show();
}

void push(int j)
{
top++;
stack[top]=j;
}

void pop()
{
stack[top]=0;
top--;
}

void show()
{
int j;
printf("\n Stack elements are: ");
```

```
for(j=0;j<10;j++)
stack[j]!=0 ?  printf(" %d", stack[j]) : printf("");
getch();
}
```

c) Program to demonstrate the operation of Stacks using Linked Lists.

```
#include <stdio.h>
#include <malloc.h>
#include<process.h>
typedef struct link_tag
{
        int data;
        struct link_tag *link;
}node;


/********** Function Declaration begins **********/
node *push(node *);
node *pop(node *);
void display(node *);
/********** Function Declaration ends **********/

void main()
{
        node *start=NULL;
        int ch;


        printf("\n\t\t Program of stack using linked list");

        do
        {
                printf("\n\t\tMenu");
                printf("\n\t\t1.Push");
                printf("\n\t\t2.Pop");
                printf("\n\t\t3.Display");
                printf("\n\t\t4.Exit");
                printf("\n\t\tEnter choice : ");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:
                                start = push(start);
                                break;
                        case 2:
                                start = pop(start);
                                break;
                        case 3:
                                printf("\n\t**** Stack *****\n");
                                display(start);
                                break;
                        case 4:
                                exit(0);
                        default:
                                printf("\nwrong choice:");
```

```
                }
        }
        while (ch!=4);
        printf("\n");
}

/********** Pushing an element in stack **********/
/********** Function Definition begins **********/
node *push(node *temp)
{
        node *new_node;
        int item;

        printf("Enter an data to be pushed : ");
        scanf("%d",&item);

        new_node = ( node *)malloc(sizeof( node));
        new_node->data = item;
        new_node->link = temp;
        temp = new_node;
        return(temp);
}
/********** Function Definition ends **********/


/********** Popping an element from stack **********/
/********** Function Definition begins **********/
node *pop(node *p)
{
        node *temp;


        if(p == NULL)
                printf("\n***** Empty *****\n");
        else
        {
                printf("Popped data = %d\n",p->data);
                temp = p->link;
                free(p);
                p = temp;
                if (p == NULL)
                        printf("\n***** Empty *****\n");
        }

        return(p);
}
/********** Function Definition ends **********/


/********** Displaying elements of Multistack1 **********/
/********** Function Definition begins **********/
```

```
void display(node *seek)
{
        printf("\nTop");
        while (seek != NULL)
        {
                printf("-> %d",seek->data);
                seek = seek->link;
        }
        printf("->NULL\n");
        return;
}
/********** Function Definition ends **********/
```

\

6 a) Program to convert a given Infix expression to Postfix.

```
/*********************************************************/
/* Program to convert an expression in Infix to Postfix.*/
/*********************************************************/

#include<stdio.h>
#include<string.h>
#include<math.h>

int top;
char stack[30];

int isp(char c)
{
        int t;
        switch(c)
        {
                case '^': t=3; break;
                case '/':
                case '*': t=2; break;
                case '+':
                case '-': t=1; break;
                case '(': t=0; break;
                default : c=1;
        }
        return(t);
}

int icp(char c)
{
        int t;
        switch(c)
        {
                case '^': t=4; break;
                case '/':
                case '*': t=2; break;
                case '+':
                case '-': t=1; break;
                case '(': t=4; break;
        }
        return(t);
}

void main()
{
        int j=0,i,l;
        char c,r,p[20]={" "},g[20];
        clrscr();
        printf("Enter the InFix Expression : ");
```

```
        gets(g);
        l=strlen(g);
        g[l]=')'; g[l+1]='$'; top++;
        stack[top]='(';
        for(i=0;g[i]!='$';i++)
        {
                c=g[i];
                if((c>='a' && c<='z') || (c>=0 && c<=9)) { j++; p[j]=c; }
                else if(c=='(') { top++; stack[top]=c; }
                else if(c==')')
                {
                        do
                        {
                                r=stack[top];
                                top--;
                                p[++j]=r;
                        }
                        while(stack[top]!='(');
                        top--;
                }
                else
                {
                        while(icp(c)<=isp(stack[top]))
                        {
                                r=stack[top];
                                top--;
                                p[++j]=r;
                        }
                        stack[++top]=c;
                }
        }
        printf("The PostFix Expression is : ");
        puts(p);
        getch();
}
```

```
/* Enter the InFix Expression : a+b/c-d
The PostFix Expression is : abc/+d-      */
```

b) Program to evaluate a given Postfix expression.

```
/**************************************/
/* Program to Evaluate Postfix Notation.*/
/**************************************/

#include<stdio.h>
#include<string.h>
void main()
{
        char s[80];
        int i,top=-1,n,x=0,y=0,stack[80];
        clrscr();
        printf("Enter the PostFix Notation : ");
        gets(s);
        n=strlen(s);
        printf("The Result of the PostFix Natation is : ");
        for(i=0;i<n;i++)
        {
          switch (s[i])
          {
            case '+' :
                        y=stack[top];
                        x=stack[top-1];
                        top=top-1;
                        x=x+y;
                        stack[top]=x;
                        break;
            case '-' :
                        y=stack[top];
                        x=stack[top-1];
                        top=top-1;
                        x=x-y;
                        stack[top]=x;
                        break;
            case '*' :
                        y=stack[top];
                        x=stack[top-1];
                        top=top-1;
                        x=x*y;
                        stack[top]=x;
                        break;
            case '/' :
                        y=stack[top];
                        x=stack[top-1];
                        top=top-1;
                        x=x/y;
                        stack[top]=x;
                        break;
            default :
```

```
                        top=top+1;
                        if(s[i]>=48 && s[i]<=65) x=s[i]-48;
                        stack[top]=x;
                        x=0;
                }
        }
        printf("%d",stack[top]);
        getch();
}

/*  Enter the PostFix Notation : 23+
The Result of the PostFix Notation is : 5  */
```

7. A) Program to explain the working of a Queue using Array.

```
/* Program: Program shows working of queue using array  */

#include<stdio.h>
#include<conio.h>
#define SIZE 20

typedef struct q_tag
{
        int front,rear;
        int item[SIZE];
}queue;

/********** Function Declaration begins **********/
void create(queue *);
void display(queue *);
void enqueue(queue *, int);
int dequeue(queue *, int);
/********** Function Declaration ends **********/

void main()
{
    int data,ch;
    queue Q;
    clrscr();
    create(&Q);
    printf("\n\t\t Program shows working of queue using array");
do
{
    printf("\n\t\t Menu");
    printf("\n\t\t 1: enqueue");
    printf("\n\t\t 2: dequeue ");
    printf("\n\t\t 3: exit. ");
    printf("\n\t\t Enter choice :");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1:
                if (Q.rear >= SIZE)
                {
                    printf("\n Queue is full");
                    continue;
                }
                else
                {
                    printf("\n Enter number to be added in a
queue");
                    scanf("%d",&data);
                    enqueue(&Q,data);
```

```
                                        printf("\n Elements in a queue are:");
                                        display(&Q);
                                        continue;
                                }
                case 2:
                                        dequeue(&Q,data);
                                        if (Q.front==0)
                                        {
                                                continue;
                                        }
                                        else
                                        {
                                                printf("\n Elements in a queue are :");
                                                display(&Q);
                                                continue;
                                        }

                case 3: printf("\n finish"); return;
        }
}while(ch!=3);
getch();
}



/********** Creating an empty queue **********/
/********** Function Definition begins **********/
void create(queue *Q)
{
        Q->front=0;
        Q->rear =0;
}
/********** Function Definition ends **********/


/********** Inserting an element in queue **********/
/********** Function Definition begins **********/
void enqueue(queue *Q, int data)
{
                if (Q->rear >= SIZE)
                {
                        printf("\n Queue is full");
                }
                if (Q->front == 0)
                {
                        Q->front = 1;
                        Q->rear = 1;
                }
                else
                {
                        Q->rear = Q->rear +1;
                }
```

```
                Q->item[Q->rear] = data;

}
/********** Function Definition ends **********/


/********** Deleting an element from queue **********/
/********** Function Definition begins **********/
int dequeue(queue *Q, int data)
{
                if (Q->front == 0)
                {
                        printf("\n Underflow.");
                        return(0);
                }
                else
                {
                        data = Q->item[Q->front];
                        printf(" \n Element %d is deleted",data);
                }

                if (Q->front==Q->rear)
                {
                        Q->front =0;
                        Q->rear = 0;
                        printf("\n Empty Queue");
                }
                else
                {
                        Q->front = Q->front +1;
                }

                return data;
}
/********** Function Definition ends **********/


/********** Displaying elements of queue **********/
/********** Function Definition begins **********/
void display(queue *Q)
{
        int x;
        for(x=Q->front;x<=Q->rear;x++)
        {
                printf("%d\t",Q->item[x]);
        }
        printf("\n\n");
}
/********** Function Definition ends **********/
```

b) Program to explain the working a Queue using linked list.

```
/* Program: Program shows working of queue using linked list */

#include <stdio.h>
#include <malloc.h>
#include<process.h>

typedef struct queue_link
{
        int data;
        struct queue_link *link;
}node;


/********** Function Declaration begins **********/
void enqueue(node **, node **, int);
void dequeue(node **);
void display(node *);
/********** Function Declaration ends **********/

void main()
{
        node *front = NULL, *rear = NULL;
        int ch,item;

        printf("\n\t\t Program of queue using linked list");

        do
        {
                printf("\n\t\tMenu");
                printf("\n\t\t1.enqueue");
                printf("\n\t\t2.dequeue");
                printf("\n\t\t3.display");
                printf("\n\t\t4.exit");
                printf("\n\t\tEnter choice : ");
                scanf("%d",&ch);

                switch(ch)
                {
                        case 1:
                                printf("Enter an data to be enqueueed : ");
                                scanf("%d",&item);
                                enqueue(&front,&rear,item);
                                break;
                        case 2:
                                dequeue(&front);
                                break;
                        case 3:
                                printf("\n\t**** Queue *****\n");
                                display(front);
```

```
                        break;
                case 4:
                        exit(0);
                default:
                        printf("\n wrong choice:");
                }
        }
        while (ch!=4);
        printf("\n");
}


/********** Inserting elements in queue **********/
/********** Function Definition begins **********/
void enqueue( node **front,node **rear,int item)
{
        node *new_node;

        new_node = (node *)malloc(sizeof( node));
        new_node->data = item;
        new_node->link = NULL;

        if ((*front) == NULL)
        {
                (*front) = new_node;
                (*rear) = new_node;
        }
        else
        {
                (*rear)->link = new_node;
                (*rear) = new_node;
        }
        return;
}

/********** Function Definition ends **********/


/********** Deleting element from queue **********/
/********** Function Definition begins **********/
void dequeue(node **front)
{
        node *temp;

        if((*front) != NULL)
        {
                temp = *front;
                (*front) = (*front)->link;
                free(temp);
        }
        return;
```

```
}
/********** Function Definition ends **********/


/********** Displaying elements of  queue **********/
/********** Function Definition begins **********/
void display(node *record)
{
        printf("\nRoot");
        while (record != NULL)
        {
                printf("-> %d",record->data);
                record = (record->link);
        }
        printf("->NULL\n");
        return;
}
/********** Function Definition ends **********/
```

8. A) Program on insertion sort

// PROGRAM ON INSERTION SORT //

```
#include<stdio.h>
void main()
{
    int A[20], N, Temp, i, j;
    clrscr();
    printf("\n\n\t ENTER THE NUMBER OF TERMS...: ");
    scanf("%d", &N);
    printf("\n\t ENTER THE ELEMENTS OF THE ARRAY...:");
    for(i=1; i<=N; i++)
    {
        gotoxy(25,11+i);
        scanf("\n\t\t%d", &A[i]);
    }
    for(i=2; i<=N; i++)
    {
        Temp = A[i];
        j = i-1;
        while(Temp<A[j] && j>=1)
        {
            A[j+1] = A[j];
            j = j-1;
        }
        A[j+1] = Temp;
    }
    printf("\n\tTHE ASCENDING ORDER LIST IS...:\n");
    for(i=1; i<=N; i++)
    printf("\n\t\t\t%d", A[i]);
    getch();
}
```

b) Program on selection sort

// Program on selection sort //

```
#include<stdio.h>
void main()
{
   int A[20], N, Temp, i, j;
   clrscr();
   printf("\n\n\t ENTER THE NUMBER OF TERMS...: ");
   scanf("%d",&N);
   printf("\n\t ENTER THE ELEMENTS OF THE ARRAY...:");
   for(i=1; i<=N; i++)
   {
       gotoxy(25, 11+i);
```

```
        scanf("\n\t\t%d", &A[i]);
    }
    for(i=1; i<=N-1; i++)
        for(j=i+1; j<=N;j++)
            if(A[i]>A[j])
            {
                Temp = A[i];
                A[i] = A[j];
                A[j] = Temp;
            }
    printf("\n\tTHE ASCENDING ORDER LIST IS...:\n");
    for(i=1; i<=N; i++)
    printf("\n\t\t\t%d",A[i]);
    getch();
}
```

c) program on shell sort.

```
// program for shell sort //

#include <stdio.h>

#define ELEMENTS 6

void shellsort(int A[],int max)
{
    int stop,swap,limit,temp,k;
  int x=(int)(max/2)-1;
  while(x>0)
  {
        stop=0;
    limit=max-x;
    while(stop==0)
    {
          swap=0;
        for(k=0;k<limit;k++)
        {
         if(A[k]>A[k+x])
        {
            temp=A[k];
          A[k]=A[k+x];
          A[k+x]=temp;
          swap=k;
        }
      }
      limit=swap-x;
      if(swap==0)
          stop=1;
    }
```

```
      x=(int)(x/2);
    }
}

int main()
{
  int i;
  int X[ELEMENTS]={5,2,4,6,1,3};
  printf("Unsorted Array:\n");
  for(i=0;i<ELEMENTS;i++)
          printf("%d ",X[i]);

  shellsort(X,ELEMENTS);
  printf("\nSORTED ARRAY\n");
  for(i=0;i<ELEMENTS;i++)
          printf("%d ",X[i]);
return ();
}
```

d) program on quick sort.

```
// program on quick sort //

#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

void main()
{
          int a[100],n,i,j,l=0,r;
          clrscr();
          printf("enter the numbwrs");
          scanf("%d",&n);
          printf("\n enter %d numbers",n);
          for(i=0;i<n;i++)
                    scanf("%d",&a[i]);
          printf("\n numbers before sorting");
          for(i=0;i<n;i++)
                    printf("%d",a[i]);
          r=n-1;
          quicksort(a,l,r);
          printf("numbers after sorting");
          for(i=0;i<n;i++)
                    printf("%d",a[i]);
          getch();
}
quicksort(int x[],int l,int r)
{
          int i,j,p,t;
          if(r>1)
```

```
{
        p=x[l];
        i=l+1;
        j=r;
}
do
{
        while(x[i]<=p && i<r)
                i++;
        while(x[j]>=p && j>l)
                j--;
        if(i<j)
        {
                t=x[i];
                x[i]=x[j];
                x[j]=t;
        }
        }
while(i<j);
        t=x[l];
x[l]=x[j];
x[j]=t;
if(j>l+1)

        quicksort(x,l,j-1);
        if(j<r-1)
        quicksort(x,j+1,r) ;
        }
```

e) Program on exchange sort.

```
// program on bubble sort //

#include<stdio.h>
#include<conio.h>

void main()
{
int a[50],t,i;
int n,c=0;
clrscr();
printf("enter the total no. of elements ");
scanf("%d",&n);
printf("enter the numbers");
for(c=0;c<n;c++)
scanf("%d",&a[c]);
c=0;
while(c<n)
{
for(i=0;i<n-c;i++)
{
```

```
if(a[i]>a[i+1])
{
t=a[i];
a[i]=a[i+1];
a[i+1]=t;
}
}
c=c+1;
}
printf("the sorted numbers are");
for(c=0;c<n;c++)
printf("%d",a[c]);
}
```

f) Program on heap sort.

```
// program on heap sort //

#include<stdio.h>
#include<conio.h>

void makeheap(int [],int );
void heapsort(int [],int );

void main()
{
        int arr[25],i,n;
        clrscr();
        printf(" heap sort");
  printf(" enter the total no. of elements : ");
  scanf("%d",&n);
printf("enter the elements of the array one by one :");
for(i=0;i<n;i++)
        scanf("%d",&arr[i]);
makeheap(arr , n );
heapsort(arr , n);
printf("after sorting");
for(i=0;i<n;i++)
        printf("%d\t", arr[i]);
getch();
}

void makeheap( int x[],int n)
{
        int i,val,s,f;
        for(i=1;i<n;i++)
        {
                val=x[i];
                s=i;
                f=(s-1)/2;
```

```
                while(s>0 && x[f]<val)
                {
                        x[s]=x[f];
                        s=f;
                        f=(s-1)/2;
                }
                x[s]=val;
                }
                }

void heapsort(int x[], int n)
{
        int i,s,f,ivalue;
        for(i=n-1;i>0;i--)
        {
                ivalue = x[i];
                x[i]=x[0];
                f=0;
                if(i==1)
                        s=-1;
                else
                        s=1;
                if(i>2 && x[2]>x[1])
                        s=2;
                while(s>=0 && ivalue<x[s])
                {
                        x[f]=x[s];
                        f=s;
                        s=2*f+1;
                        if(s+1<=i-1 && x[s]<x[s+1])
                                s++;
                        if(s>i-1)
                                s=-1;
                }
                x[f]=ivalue;
                }
                }
```

g) Program on Merge Sort.

```
// program on merge sort //

#include<stdio.h>
#include<conio.h>
void main()
{
        int a[25],b[25],c[25];
        int n1,n2,i,j,k,temp;
        clrscr();
        printf("merge sort \n");
```

```
printf(" enter the total no. of elements in the first array: ");
scanf("%d", &n1);
printf(" enter the elements of the first array one by one : ");
for(i=0;i<n1;i++)
        scanf("%d", &a[i]);
printf(" enter the total no. of elements in the second array: ");
scanf("%d", &n2);
printf(" enter the elements of the second array one by one : ");
for(i=0;i<n2;i++)
        scanf("%d", &b[i]);

for(i=0;i<n1;i++)
{
        for(j=i+1;j<n1;j++)
        {
                if(a[i]>a[j])
                {
                        temp=a[i];
                        a[i]=a[j];
                        a[j]=temp;
                }
                }
                }
for(i=0;i<n2;i++)
{

        for(j=i+1;j<n2;j++)
        {
                if(b[i]>b[j])
                {
                        temp=b[i];
                        b[i]=b[j];
                        b[j]=temp;
                }
                }
                }
for(i=j=k=0;i<(n1+n2); )
{
        if(a[j]<=b[k])
                c[i++]=a[j++];
        else
                c[i++]=b[k++];
        if(j==n1 || k==n2)
                break;
}
for( ; j<=n1; )
        c[i++]=a[j++];
for( ;k<=n2; )
        c[i++]=b[k++];
printf("\n array after sorting: \n");
for(i=0;i<(n1+n2);i++)
```

```
                printf("%d\t", c[i]);
        getch();
        }
```

9. Program on trees

```
#include<stdio.h>
# define NULL 0
struct node
{
        int data;
        struct node *left;
        struct node *right;
}*head,*prev,*temp,*newnode,*p1;
void main()
{
        int d;
        do{
                clrscr();
                printf("\n Tree Operation");
                printf("\n 1.Create");
                printf("\n 2.Traverse");
                printf("\n 3.Exit");
                printf("\n Enter your choice");
                scanf("%d",&d);
                switch(d)
                {
                        case 1: create();
                                break;
                        case 2: traverse();
                                break;
                }
        }while(d!=3);
}


create()
{
int n,i;
printf("Enter how many nodes you want to insert\n");
scanf("%d",&n);
head=NULL;
printf("Enter the data");
for(i=1;i<=n;i++)
{
        newnode=(struct node *) malloc(sizeof(struct node));
        temp=head;
        scanf("%d",&newnode->data);
        newnode->left=NULL;
        newnode->right=NULL;
```

```
        if(head==NULL)
                head=newnode;
        else
        {
                while(temp!=NULL)
                {
                        prev=temp;
                        if(newnode->data<temp->data)
                                temp=temp->left;
                        else
                                temp=temp->right;
                }

                if(newnode->data<prev->data)
                        prev->left=newnode;
                else
                        prev->right=newnode;
        }
}
return;
}

traverse()
{
int d;
do{
        clrscr();
        printf("\nTraversal Operations");
        printf("\n1. In Order");
        printf("\n2. PreOrder");
        printf("\n3. PostOrder");
        printf("\n4. Exit");
        printf("\nEnter your chioce");
        scanf("%d",&d);
        switch(d)
        {
                case 1:
                        printf("The inorder is::\n");
                        inorder();
                        getch();
                        break;
                case 2:
                        printf("The Preorder is::\n");
                        preorder();
                        getch();
                        break;
                case 3:
                        printf("The postorder is ::\n");
                        postorder();
                        getch();
                        break;
        }
```

```
}while(d!=4);
return;
}

inorder()
{
        int top=0;
        int stack[100],pp;
        struct node *p1;
        stack[top]=NULL;
        p1=head;
        do{
                while(p1!=NULL)
                {
                        top=top+1;      //PUSH OPERATION
                        stack[top]=p1;
                        p1=p1->left;
                }
                if(top!=0)
                {
                        p1=stack[top];   //POP OPERATION
                        top--;
                        printf("%5d",p1->data);
                        p1=p1->right;
                }
        }while((p1!=NULL) || (top!=0));
        return;
}
preorder()
{
        int top=0;
        int stack[100];
        struct node *p1;
        stack[top]=NULL;
        p1=head;
        do{
                while(p1!=NULL)             //PUSH OPERATION
                {
                        printf("%5d",p1->data);
                        top++;
                        stack[top]=p1;
                        p1=p1->left;
                }
                p1=stack[top];
                top--;
                p1=p1->right;
        }while((top!=0) || (p1!=NULL));
        return;
}
```

```
postorder()
{
int top=0,c=0;
int stack[100],pp,temp[20],i,j;
struct node *p1;
stack[top]=NULL;
p1=head;
do{
        while(p1!=NULL)
        {
                top++;
                stack[top]=p1;
                if(p1->right!=NULL)
                {
                        stack[++top]=p1->right;
                        stack[top]=-stack[top];
                }
                p1=p1->left;
        }
        pp=stack[top];
        top--;
        while(pp>0)
        {
                p1=pp;
                temp[c++]=p1->data;
                pp=stack[top--];
        }
        if (pp<0) p1=-pp;
}while(top!=NULL);
for(i=0;i<c;i++)
{
        for(j=0;j<c;j++)
        {
        if ((i!=j) && (temp[i]==temp[j]))
                temp[j]=0;
        }
}
        for(i=0;i<c;i++)
        {
                if (temp[i]!=0)
                        printf("%5d",temp[i]);
        }
        return;
}
```

10. Program on Hashing

```
# include <stdio.h>
# include <conio.h>
# include <string.h>
# include <stdlib.h>
# define RECSIZE 12

void curr_pos(int r,int c)
{
    char cmd[15];
    sprintf(cmd,"tput cup %d%d",r,c);
    system(cmd);
}

void disp_scr()
{
    curr_pos(2,5);
    printf("Data Entry Screen\n");
    curr_pos(4,5);
    printf("S.No.\n");
    curr_pos(4,15);
    printf("City Name\n");
    curr_pos(4,45);
    printf("Population\n");
}

isalfa(char);
void get_val_city(char str[],int r,int c)
{
    int  valid=0;
    char  *temp;
    while(!valid)
    {
            curr_pos(r,c);
            gets(str);
            fflush(stdin);
            if(strlen(str) > 10)
                    valid=0;
            else
            {
                    valid=1;
                    temp=str;
                    while(*temp && valid)
                    {
                            if(!isalfa(*temp++))
                            valid=0;
                    }
            }
    }
    curr_pos(20,8);
}
```

```
isalfa(char c)
{
    return((((c>='A') && (c<='Z')) || ((c>='a') && (c<='z')))?1:0);
}

void get_val_pop(char str[],int r,int c)
{
    int valid=0,len;
    char temp[3];
    while(!valid)
    {
            curr_pos(r,c);
            gets(str);
            if (((len=strlen(str)) > 2) || (len==0))
                    valid=0;
            else if(atoi(str) > 0)
            {
                    valid =1;
                    sprintf(temp,"%202s",str);
                    strcpy(str,temp);
            }
    }
    curr_pos(20,8);
}

void cr_fl_spc(FILE *fp,int nbuck,int recl)
{
    int num, spaces;
    spaces=nbuck*recl;
    for(num=0;num<spaces;num++)
            fputc(' ',fp);
}

get_hashno(char *key_val)
{
    if (key_val[0]>='a')
            return(key_val[0]-'a');
    else if (key_val[0]>='A');
            return(key_val[0]-'A');
}


void put_record(int hno,FILE *fp,char *fld1,char *fld2,int recl)
{
    fseek(fp,(long)(recl*hno),0);
    fputs(fld1,fp);
    fputs(fld2,fp);
}
```

```
void main()
{
    char city[11],temp_city[11],pop[3],more='y';
    int row,col,hash_no,sno=0;
    FILE *fp;
    fp=fopen("cityinfo.txt","w");
    if((fp=fopen("cityinfo.txt","w"))==NULL)
    {
            printf("Error opening file");
            exit(0);
    }

    system("tput clear");
    disp_scr();

    cr_fl_spc(fp,28,RECSIZE);
    row=6;
    col=6;
    while(more=='y')
    {
    curr_pos(row,col);
            printf("%d\n",++sno);
            get_val_city(city,row,col+10);
            get_val_pop(pop,row,col+40);
            if(city!='\0')
            {
                    hash_no=get_hashno(city);
                    sprintf(temp_city,"%10.10s",city);
                    put_record(hash_no,fp,city,pop,12);
            }
            more='\0';
            while(more!='y' && more!='n')
            {
                    curr_pos(20,8);
                    printf("Any more Records?(y/n):");
                    more=getchar();
                    fflush(stdin);
            }
            curr_pos(20,8);
            printf("\n");
            row++;
    }
    fclose(fp);
}
```

11. Program on Binary Search Tree

```
/*  program to build binary search tree from array. */

#include<stdio.h>
#include<conio.h>
#include<alloc.h>

struct node
{
     struct node *left ;
     char data ;
     struct node *right ;
} ;
struct node *root;
struct node * buildtree(int);
char arr[ ] = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', '\0', '\0', 'H' } ;
int  lc[ ] = {  1,  3,  5,  -1,  9, -1, -1,  -1,  -1, -1 } ;
int  rc[ ] = {  2,  4,  6,  -1, -1, -1, -1,  -1,  -1, -1 } ;
void display();
void main( )
{
 int ch;
  clrscr();
  do
  {
  printf("1. Create. .....\n");
  printf("2. Display .....\n");
  printf("3. Exit. .....\n");
  printf("enter your choice(1..3)\n");
  scanf("%d",&ch);
  switch(ch)
  {
  case 1:
  root = buildtree(0);
  break;
  case 2:
  display();
  break;
  default :
  break;
  }
  getch();
  } while(ch != 3);
  }
  void display()
  {
  void inorder(struct node *);
```

```
void preorder(struct node
*);void postorder(struct
node *);int ch;
do
{
printf("1. Inorder……\n");
printf("2. Preorder…..\n");
printf("3. Postorder. .. \n");
printf("4. Exit……..\n");
printf("enter your choice(1. 4)");
scanf("%d",&ch);
switch(ch)
{
case 1:
inorder(root);
break;
case 2:
preorder(root);
break;
case 3:
postorder(root);
break;
default :
break;
}
getch();
} while(ch !=4);
}

struct node *buildtree(int index)
{
struct node *temp =
NULL;if(index != -1)
{
temp=(struct node *)malloc(sizeof(struct
node));temp->left=buildtree(lc[index]);
temp->data=arr[index];
temp->right=buildtree(rc[index]);
}
return temp;
}
void inorder(struct node *root)
{
if(root != NULL)
{
inorder(root->left);
printf("%c\t", root-
>data);inorder(root-
>right);
}
}
```

```
void preorder(struct node *root)
{
if(root != NULL)
{
printf("%c\t", root->data);
preorder(root->left);
preorder(root->right);
}
}
void postorder(struct node *root)
{
if(root != NULL)
{
preorder(root->left);
postorder(root->right);
printf("%c\t",root->data);
}
}
```

# LAB MANUAL

# NETWORK PROGRAMMING

# COS-192

# OBJECTIVES

This course provides students with hands on training regarding the design, troubleshooting, modeling and evaluation of computer networks. In this course, students are going toexperiment in a real test-bed networking environment, and learn about network design and troubleshooting topics and tools such as: network addressing, Address Resolution Protocol (ARP), basic troubleshooting tools (e.g. ping, ICMP), IP routing (e,g, RIP), route discovery(e.g. traceroute), TCP and UDP, IP fragmentation and many others. Student will also be introduced to the network modeling and simulation, and they will have the opportunity to build some simple networking models using the tool and perform simulations that will help them evaluate their design approaches and expected network performance.

# CONTENT

Ex. No. 1.a

Date:

# Looking up internet address of local host

Aim

To find the IP address of a local host using java program

Theory

## Inet Address

public class InetAddress
extends Object
implements Serializable

It is a class in Java which represents an Internet Protocol (IP) address. An instance of an InetAddress consists of an IP address and possibly corresponding hostname. The Class represents an Internet address as Two fields: 1- Host name (The String)=Contain the name of the Host.2- Address(an int)= The 32 bit IP address. These fields are not public, so we can't Access them directly. There is not public constructor in InetAddress class, but it has 3 static methods that returns suitably initialized InetAddress Objects namely Public String getHostName() , public byte[] getAddress() and public String getHostAddress()

## getLocalHost()

public static InetAddress getLocalHost()
throws UnknownHostException

Returns the address of the local host. This is achieved by retrieving the name of the host from the system, then resolving that name into an InetAddress.

Note: The resolved address may be cached for a short period of time.

If there is a security manager, its checkConnect method is called with the local host name and -1 as its arguments to see if the operation is allowed. If the operation is not allowed, an InetAddress representing the loopback address is returned.

Returns:
the address of the local host.
Throws:
UnknownHostException - if the local host name could not be resolved into an address.

SOURCE CODE:

```
import java.util.*;
import java.lang.*;
import java.net.*;

public class getOwnIP
{
 public static void main(String args[])throws UnknownHostException
 {
  try
  {
   InetAddress     IPO=InetAddress.getLocalHost();
   System.out.println("IP of this system="+IPO.getHostAddress());
  }
  catch(Exception e)
  {
   System.out.println("Exception        caught="+e.getMessage());
  }
 }
}
```

SAMPLE INPUT/OUTPUT:

```
E:\EX1>javac getOwnIP.java
E:\EX1>java getOwnIP
IP of this system=10.1.60.11
```

Ex. No. 1.b

Date:

# Looking up internet address of Remote host

Aim

To find the IP address of a remote host using java program

Theory

1. getByName

public static InetAddress getByName(String host)
throws UnknownHostException

Determines the IP address of a host, given the host's name.

The host name can either be a machine name, such as "java.sun.com", or a textual representation of its IP address. If a literal IP address is supplied, only the validity of the address format is checked.

For host specified in literal IPv6 address, either the form defined in RFC 2732 or the literal IPv6 address format defined in RFC 2373 is accepted. IPv6 scoped addresses are also supported. See here for a description of IPv6 scoped addresses.

Parameters:
host - the specified host, or null.

Returns:
an IP address for the given host name.

Throws:
UnknownHostException - if no IP address for the host could be found, or if a scope_id was specified for a global IPv6 address.

SecurityException - if a security manager exists and its checkConnect method doesn't allow the operation

SOURCE CODE:

```
import java.util.*;
import java.lang.*;
import java.net.*;

public class getRemoteIP
{
 public static void main(String args[])
 {
  try
  {
   InetAddress     IPO=InetAddress.getByName(args[0]);
   System.out.println("IP of this system = " +IPO);
  }
  catch(Exception e)
  {
   System.out.println("Exception caught = "+e.getMessage());
  }
 }
}
```

SAMPLE INPUT/OUTPUT:

```
E:\EX1>javac getRemoteIP.java
E:\EX1>java getRemoteIP
IP of this system = GFL-335/10.1.60.11
```

Ex. No. 2.

Date:

# Implementation of port scanner

Aim

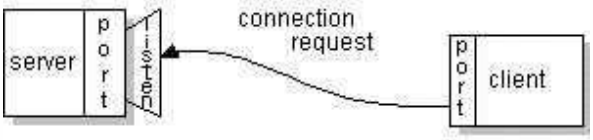
To implement the port scanner using java program

Theory

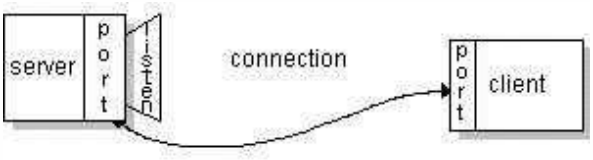
Socket

Normally, a server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request.

On the client-side: The client knows the hostname of the machine on which the server is running and the port number on which the server is listening. To make a connection request, the client tries to rendezvous with the server on the server's machine and port. The client also needs to identify itself to the server so it binds to a local port number that it will use during this connection. This is usually assigned by the system.




If everything goes well, the server accepts the connection. Upon acceptance, the server gets a new socket bound to the same local port and also has its remote endpoint set to the address and port of the client. It needs a new socket so that it can continue to listen to the original socket for connection requests while tending to the needs of the connected client.




On the client side, if the connection is accepted, a socket is successfully created and the client can use the socket to communicate with the server.

The client and server can now communicate by writing to or reading from their sockets.

---

Definition:

A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to.

An endpoint is a combination of an IP address and a port number. Every TCP connection can be uniquely identified by its two endpoints. That way you can have multiple connections between your host and the server.

The java.net package in the Java platform provides a class, Socket, that implements one side of a two-way connection between your Java program and another program on the network. The Socket class sits on top of a platform-dependent implementation, hiding the details of any particular system from your Java program. By using the java.net.Socket class instead of relying on native code, your Java programs can communicate over the network in a platform- independent fashion.

Additionally, java.net includes the ServerSocket class, which implements a socket that servers can use to listen for and accept connections to clients. This lesson shows you how to use the Socket and ServerSocket classes.

Socket constructor

public Socket(InetAddress address, int port) throws IOException

Creates a stream socket and connects it to the specified port number at the specified IP address.

If the application has specified a socket factory, that factory's createSocketImpl method is called to create the actual socket implementation. Otherwise a "plain" socket is created.

If there is a security manager, its checkConnect method is called with the host address and port as its arguments. This could result in a SecurityException.

Parameters:
address - the IP address.
port - the port number.

Socket close method

public void close() throws IOException
Closes this socket.

Any thread currently blocked in an I/O operation upon this socket will throw a SocketException.

Once a socket has been closed, it is not available for further networking use (i.e. can't be reconnected or rebound). A new socket needs to be created.

Closing this socket will also close the socket's InputStream and OutputStream.

SOURCE CODE:

```
import java.net.*;

import java.io.*;


public class PortScanner

{

 public static void main(String args[])

 {

  int  startPortRange  =  0;

  int  stopPortRange  =  0;

  int n=1;

  startPortRange = Integer.parseInt(args[0]);

  stopPortRange = Integer.parseInt(args[1]); for(int

  i=startPortRange; i<=stopPortRange; i++)

  {

   try

   {

    Socket Serversok=new Socket("127.0.0.1",i);

    System.out.println("Port in use : "+i);

    Serversok.close();

    n=0;

   }

  catch(Exception e)

   {}
```

```
  if(n!=0)
     System.out.println("Port not in use : "+i);
   n=1;
  }
 }
}
```

SAMPLE INPUT/OUTPUT:

E:\EX2>javac PortScanner.java

E:\EX2>java PortScanner 132 137

Port not in use : 132

Port not in use : 133

Port not in use : 134

Port not use : 135

Port not in use : 136

Port not in use : 137

E:\EX2>java PortScanner 442 447

Port not in use : 442

Port not in use : 443

Port not in use : 444

Port not use : 445

Port not in use : 446

Port not in use : 447

Ex. No. 3.

Date:

# Implementation of finger client

Aim

To implement the finger client using java program

Theory

## Socket

Sockets provide the communication mechanism between two computers using TCP. A client program creates a socket on its end of the communication and attempts to connect that socket to a server.

When the connection is made, the server creates a socket object on its end of the communication. The client and server can now communicate by writing to and reading from the socket.

## Class Writer

public abstract class Writer extends Object implements Appendable, Closeable, Flushable

Abstract class for writing to character streams. The only methods that a subclass must implement are write(char[], int, int), flush(), and close(). Most subclasses, however, will override some of the methods defined here in order to provide higher efficiency, additional functionality, or both.

## Methods

InputStream getInputStream()
Returns an input stream for this socket.

OutputStream getOutputStream()
Returns an output stream for this socket.

void close()
Closes this socket.

SOURCE CODE FOR CLIENT:

```
importjava.io.BufferedInputStream; import
java.io.InputStream;
importjava.io.InputStreamReader;
importjava.io.OutputStreamWriter; import
java.io.Writer;
import java.net.Socket;


public class Main
{
 public final static int DEFAULT_PORT=79;
 public static void main(String args[]) throws Exception
 {
  Stringhostname="localhost";
  Socket connection=null;
  connection=new       Socket(hostname,DEFAULT_PORT);
  Writerout=new OutputStreamWriter(connection.getOutputStream(),"8859_1"); out.write("\r\n");
  InputStream     raw=connection.getInputStream();
  BufferedInputStreambuffer=new BufferedInputStream(raw);
  InputStreamReaderin=newInputStreamReader(buffer,"8859_1"); int c;
  while((c=in.read())!=-1)
  {
```

```
  if((c>=32 && c<127)||c=='\t'||c=='\r'||c=='\n')
   {
    System.out.write(c);
   }
  }
  connection.close();
 }
}
```

SOURCE CODE FOR SERVER:

```
importjava.io.BufferedOutputStream;
importjava.io.BufferedReader; import
java.io.FileReader;
importjava.io.InputStreamReader; import
java.io.PrintWriter;
importjava.net.InetSocketAddress;
import java.net.ServerSocket; import
java.net.Socket;
importjava.nio.channels.SelectionKey;
importjava.nio.channels.Selector;
import     java.nio.channels.ServerSocketChannel;
```

```
import java.util.Iterator;
import java.util.Set;


public class New
{
 public static void readPlan(String userName,PrintWriter pw)throws Exception
 {
  FileReaderfile=newFileReader(userName+".plan");
  BufferedReader buff=new BufferedReader(file);
  boolean eof=false;
  pw.println("\n userName:"+userName +"\n");
  while(!eof)
  {
   Stringline=buff.readLine();
   if(line==null)
   eof=true;
   else
   pw.println(line);
  }
  buff.close();
 }
 public static void main(String args[])throws Exception
 {
  ServerSocketChannel        sockChannel=ServerSocketChannel.open();
  sockChannel.configureBlocking(false);
```

```
InetSocketAddressserver=newInetSocketAddress("localhost",79);
ServerSocket socket=sockChannel.socket();
socket.bind(server);
Selectorselector=Selector.open();
sockChannel.register(selector,SelectionKey.OP_ACCEPT);
while(true)
{
 selector.select();
 Setkeys=selector.selectedKeys(); Iterator
 it=keys.iterator(); while(it.hasNext())
 {
  SelectionKey selKey=(SelectionKey)it.next();
  it.remove();
  if(selKey.isAcceptable())
  {
   ServerSocketChannel selChannel=(ServerSocketChannel)selKey.channel();
   ServerSocket selSocket=selChannel.socket();
   Socketconnection=selSocket.accept();
   InputStreamReaderisr=new InputStreamReader(connection.getInputStream());
   BufferedReader is=new BufferedReader(isr);
   PrintWriter pw=new PrintWriter(new
BufferedOutputStream(connection.getOutputStream()),false);
   pw.println("NIO finger server");
   pw.flush();
```

```
    String outLine=null;
    StringinLine=is.readLine();
    if(inLine.length()>0)
    {
     outLine=inLine;
    }
    readPlan(outLine,pw);
    pw.flush();
    pw.close();
    is.close();
    connection.close();
   }
  }
 }
}
```

SAMPLE INPUT/OUTPUT:

E:\EX3>javac New.java

E:\EX3>java New

E:\EX3>javac Main.java

E:\EX3>java Main

NIO finger server

Ex. No. 4.

Date:

# Implementation of ping programming

Aim

To implement the ping programming using java program

Theory

## Inetaddress

This class represents an Internet Protocol (IP) address.

## getByName

public static InetAddress getByName(String host)
throws UnknownHostException Determines
the IP address of a host, given the host's name.

The host name can either be a machine name, such as "java.sun.com", or a textual representation of its IP address. If a literal IP address is supplied, only the validity of the address format is checked.

Parameters:
host - the specified host, or null.
Returns:
an IP address for the given host name.

## isReachable

public boolean isReachable(NetworkInterface netif, int
ttl,
int timeout)
throws IOException

Test whether that address is reachable. Best effort is made by the implementation to try to reach the host, but firewalls and server configuration may block requests resulting in a unreachable status while some specific ports may be accessible. A typical implementation will use ICMP ECHO REQUESTs if the privilege can be obtained, otherwise it will try to establish a TCP connection on port 7 (Echo) of the destination host.

The network interface and ttl parameters let the caller specify which network interface the test will go through and the maximum number of hops the packets should go through. A negative value for the ttl will result in an IllegalArgumentException being thrown.

The timeout value, in milliseconds, indicates the maximum amount of time the try should take. If the operation times out before getting an answer, the host is deemed unreachable. A negative value will result in an IllegalArgumentException being thrown.

Parameters:

netif - the NetworkInterface through which the test will be done, or null for any interface ttl - the maximum numbers of hops to try or 0 for the default timeout - the time, in milliseconds, before the call aborts

Returns:

a Boolean indicating if the address is reachable.

SOURCE CODE:

```
import java.net.*;

import java.io.*;

public class Ping

{

 public static void main(String args[])

 {

  System.out.println("Pingingstatus");

  String ipa="GFL-335";

  try

  {

   InetAddress      IPA=InetAddress.getByName("Gfl-335");

   System.out.println("Sending ping request to " +ipa);

   booleanstatus=IPA.isReachable(50000);

   if(status)

   {

    System.out.println("Status : Host is reachable");

   }

   else

   {

    System.out.println("Status : Host is not reachable");
```

```
}
 }

 catch(IOException e)
 {
  System.out.println("Host does not exist");
 }
}
}
```

SAMPLE INPUT/OUTPUT:

E:\EX4>javac Ping.java

E:\EX4>java Ping Pinging

status

Sending ping request to GFL-335

Status : Host is reachable

Ex. No. 5.

Date:

# Implementation of peer to peer communication using UDP

Aim

To implement the peer to peer communication through UDP using java program

Theory

Peer to peer communication

P2P, P-to-P and P2P communications, peer-to-peer communication refers to the transmission between two peer computers over a network. P2P became widely known by computer users as they began sharing MP3s and other files over P2P networks

UDP

UDP (User Datagram Protocol) is a communications protocol that offers a limited amount of service when messages are exchanged between computers in a network that uses the Internet Protocol (IP).

UDP Datagram

UDP network traffic is organized in the form of datagrams . A datagram comprises one message unit. The first eight (8) bytes of a datagram contain header information and the remaining bytes contain message data.

A UDP datagram header consists of four (4) fields of two bytes each:

- source port number
- destination port number
- datagram size
- checksum

UDP port numbers allow different applications to maintain their own channels for data similar to TCP. UDP port headers are two bytes long; therefore, valid UDP port numbers range from 0 to 65535.

The UDP datagram size is a count of the total number of bytes contained in header and data sections. As the header length is a fixed size, this field effectively tracks the length of the variable-sized data portion (sometimes called payload). The size of datagrams varies depending on the operating environment but has a maximum of 65535 bytes.

UDP checksums protect message data from tampering. The checksum value represents an encoding of the datagram data calculated first by the sender and later by the receiver. Should an individual datagram be tampered with or get corrupted during transmission, the UDP protocol detects a checksum calculation mismatch. In UDP, checksumming is optional as opposed to TCP where checksums are mandatory.

Datagram Socket

public class DatagramSocket
extends Object implements
Closeable

This class represents a socket for sending and receiving datagram packets.

A datagram socket is the sending or receiving point for a packet delivery service. Each packet sent or received on a datagram socket is individually addressed and routed. Multiple packets sent from one machine to another may be routed differently, and may arrive in any order.

BufferedReaderclass

The BufferedReader class is used for fast reading operations of texts from a character- input stream. It can be used to read single characters, arrays, and lines of data. The size of buffer may or may not be specified. The readLine() method of the BufferedReader class can be used to get the next line of characters from a file, and the skip(long n) method can be used to skip n number of characters.

SOURCE CODE FOR CLIENT:

```
import java.io.*; import
java.net.*;
class UDPClient
{
 public static void main(String args[])throws IOException
 {
  BufferedReaderinFromUser=newBufferedReader(newInputStreamReader(System.in));
  DatagramSocket clisock=newDatagramSocket();
  InetAddress     IPA=InetAddress.getByName("GFL-335");  byte[]
```

```
 receivedata=new byte[1024]; byte[]
 senddata=new byte[1024];
 String   sentence=inFromUser.readLine();  senddata
 =sentence.getBytes();
 DatagramPacket     sendpack=new     DatagramPacket(senddata,senddata.length,IPA,9876);
 clisock.send(sendpack);
 DatagramPacketrecpack=new DatagramPacket(receivedata,receivedata.length);
 clisock.receive(recpack);
 String       msentence=new       String(recpack.getData());
 System.out.println("From   Server   :   "+   msentence);
 clisock.close();
 }
}
```

SOURCE CODE FOR SERVER:

```
import java.io.*; import
java.net.*;


class UDPServer
{
 public static void main(String args[])throws IOException
 {
 DatagramSocketsersock=newDatagramSocket(9876);
 byte[] receivedata=newbyte[1024];
 byte[] senddata=new byte[1024];
 while(true)
```

```
DatagramPacketrecpack=new DatagramPacket(receivedata,receivedata.length); sersock.receive(recpack);
      String sentence=new String(recpack.getData());
      System.out.println("Received : "+ sentence); InetAddress
      IPA=recpack.getAddress();
      int port=recpack.getPort();
      String    csentence=sentence.toUpperCase();
      senddata=csentence.getBytes();
      DatagramPacket     sendpack=new    DatagramPacket(senddata,senddata.length,IPA,port);
      sersock.send(sendpack);
     }
    }
   }
```

SAMPLE INPUT/OUTPUT:

```
E:\EX5>javac UDPServer.java

E:\EX5>java UDPServer
Received : Hi Server

E:\EX5>javac UDPClient.java

E:\EX5>java UDPClient Hi Server
From Server : HI SERVER
```

Ex. No. 6.

Date:

# Implementation of socket program for UDP Echo Client and Echo Server

Aim

To implement a socket program for UDP Echo Client and Echo Server using java program

Theory

DatagramSocket

public class DatagramSocket
extends Object implements
Closeable

This class represents a socket for sending and receiving datagram packets.

A datagram socket is the sending or receiving point for a packet delivery service. Each packet sent or received on a datagram socket is individually addressed and routed. Multiple packets sent from one machine to another may be routed differently, and may arrive in any order.

BufferedReaderclass

The BufferedReader class is used for fast reading operations of texts from a character- input stream. It can be used to read single characters, arrays, and lines of data. The size of buffer may or may not be specified. The readLine() method of the BufferedReader class can be used to get the next line of characters from a file, and the skip(long n) method can be used to skip n number of characters.

SOURCE CODE FOR CLIENT:

```
import java.net.*;
import java.io.*;
import java.lang.*;

public class EC
{
 public static void main(String args[])throws IOException
 {
  byte[] buff=new byte[1024];
  DatagramSocketsoc=newDatagramSocket(9999);
  String s="From client-Hello Server";
  buff=s.getBytes();
  InetAddress     a=InetAddress.getByName("Gfl-335");
  DatagramPacketpac=newDatagramPacket(buff,buff.length,a,8888);
  soc.send(pac);
  System.out.println("End of sending");
  byte[] buff1=new byte[1024];
  buff1=s.getBytes();
  pac=new     DatagramPacket(buff1,buff1.length);
  soc.receive(pac);
  String msg=new String(pac.getData());
  System.out.println(msg);
```

```
 System.out.println("End of programming");
 }
}
```

SOURCE CODE FOR SERVER:

```
import java.net.*;
import java.io.*;
import java.lang.*;

public class ES
{
 public static void main(String args[])throws IOException
 {
  byte[] buff=new byte[512];
  DatagramSocket soc=new DatagramSocket(8888);
  DatagramPacket pac=new DatagramPacket(buff,buff.length);
  System.out.println("serverstarted");
  soc.receive(pac);
  String msg=new String(pac.getData());
  System.out.println(msg);
  System.out.println("End of reception");
```

```
 String s="From Server-Hello client"; byte[] buff1=new
 byte[512]; buff1=s.getBytes(); InetAddress
 a=pac.getAddress(); int port=pac.getPort();
 pac=new     DatagramPacket(buff,buff1.length,a,port);
 soc.send(pac);
 System.out.println("End of sending");
 }
}
```

SAMPLE INPUT/OUTPUT:

```
E:\EX6>javac ES.java

E:\EX6>java ES server
started
From client-Hello Server

End of reception End of sending

E:\EX6>javac EC.java

E:\EX6>java EC

End of sending
From client-Hello Server End of
programming
```

Ex. No. 7.

Date:

# Implementation of Client Server Communication Using TCP

Aim

To implement Client Server Communication using TCP through Java programming.

Theory

DatagramSocket

public class DatagramSocket
extends Object implements
Closeable

This class represents a socket for sending and receiving datagram packets.

A datagram socket is the sending or receiving point for a packet delivery service. Each packet sent or received on a datagram socket is individually addressed and routed. Multiple packets sent from one machine to another may be routed differently, and may arrive in any order.

BufferedReaderclass

The BufferedReader class is used for fast reading operations of texts from a character- input stream. It can be used to read single characters, arrays, and lines of data. The size of buffer may or may not be specified. The readLine() method of the BufferedReader class can be used to get the next line of characters from a file, and the skip(long n) method can be used to skip n number of characters.

TCP

TCP is one of the main protocols in TCP/IP networks. Whereas the IP protocol deals only with packets, TCP enables two hosts to establish a connection and exchange streams of data. TCP guarantees delivery of data and also guarantees that packets will be delivered in the same order in which theywere sent.

SOURCE CODE FOR CLIENT:

```
import java.lang.*;
import java.net.*;
import java.io.*;

class CliTCP
{
 public static void main(String args[])
 {
  try
  {
   Socket skt=new Socket("Gfl-335",1234);
   BufferedReaderin=new BufferedReader(new InputStreamReader(skt.getInputStream()));
   System.out.println("Receivedstring:");
   while(!in.ready())
   {
   }
   System.out.println(in.readLine());
   System.out.println("\n"); in.close();
  }
  catch(Exception e)
  {
   System.out.println("Whoops! It didn't work! \n");
```

```
 }
 }
}
```

SOURCE CODE FOR SERVER

```
import java.lang.*;
import java.net.*;
import java.io.*;

class SerTCP
{
 public static void main(String args[])
 {
  String data="Welcome";
  try
  {
   ServerSocket s=new ServerSocket(1234); Socket
   skt=s.accept(); System.out.println("Server has
   connected! \n");
   PrintWriterout=new PrintWriter(skt.getOutputStream(),true);
   System.out.println("Sending string: \n "+data+"\n");
```

```
   out.print(data);
   out.close();
   skt.close();
   s.close();
  }
  catch(Exception e)
  {
   System.out.println("Whoops! It didn't work! \n");
  }
 }
}
```

SAMPLE INPUT/OUTPUT:

```
E:\EX7>javac SerTCP.java

E:\EX7>java SerTCP
Server has connected!

Sending string:
 Welcome

E:\EX7>javac CliTCP.java


E:\EX7>java CliTCP


Received string:
Welcome
```

Ex. No. 8.

Date:

# Implementation of Client server Application for chat

Aim

To write a java program to implement client server application for chat.

Theory

Inet Address

```
public class InetAddress
                    extends Object
                    implements Serializable
```

It is a class in Java which represents an Internet Protocol (IP) address. An instance of an InetAddress consists of an IP address and possibly corresponding hostname. The Class represents an Internet address as Two fields: 1- Host name (The String)=Contain the name of the Host.2- Address(an int)= The 32 bit IP address. These fields are not public, so we can't Access them directly. There is not public constructor in InetAddress class, but it has 3 static methods that returns suitably initialized InetAddress Objects namely Public String getHostName() , public byte[] getAddress() and public String getHostAddress()

BufferedReaderclass

The BufferedReader class is used for fast reading operations of texts from a character- input stream. It can be used to read single characters, arrays, and lines of data. The size of buffer may or may not be specified. The readLine() method of the BufferedReader class can be used to get the next line of characters from a file, and the skip(long n) method can be used to skip n number of characters.

SOURCE CODE FOR CLASS CHAT:

```
import java.io.*;
import java.net.*;
importjava.lang.String.*;

class chat extends Thread
{
 Socket soc;
 InetAddress addr;
 ServerSocket s;
 BufferedReader d;
 BufferedReader in;
 PrintWriter out; String
 name;
 public chat(Socket s)throws IOException
 {
  soc= s;
  d=new BufferedReader(new InputStreamReader(System.in));
  in=new BufferedReader(new InputStreamReader(soc.getInputStream()));
  out=new PrintWriter(new BufferedWriter(new
OutputStreamWriter(soc.getOutputStream())),true);
  start();
 }
```

```
public void run()
{
 String str; try
 {
  out.println("Chat sessions begins..");
  out.println("Server: Your name please : ");
  str=in.readLine();
  name=str;
  addr=soc.getInetAddress();
  System.out.println("Message:");
  str=d.readLine();
  while(true)
  {
   out.println(str); str=in.readLine();
   if(str.equalsIgnoreCase("end"))
   break;
   System.out.println(name+":>"+str);
   System.out.println("Message : ");
   str=d.readLine();
  }
 }
 catch(IOException e)
 {}
```

```
 catch(NullPointerException e)
 {
  System.out.println(name+"quitchat");
 }
}
}
```

SOURCE CODE FOR CHAT CLIENT:

```
import java.io.*;
import java.net.*;

class chatclient
{
 public static Socket soc;
 public static void main(String args[])throws IOException
 {
  try
  {
   InetAddress     a=InetAddress.getLocalHost();
   soc=new Socket(a,0202);
   BufferedReader d=new BufferedReader(new InputStreamReader(System.in));
```

```
BufferedReaderin=new BufferedReader(new InputStreamReader(soc.getInputStream()));
PrintWriterout=newPrintWriter(newBufferedWriter(newBufferedWriter(new
OutputStreamWriter(soc.getOutputStream()))),true);
String s;
s=in.readLine();
System.out.println(s);
s=in.readLine();
System.out.println(s);
s=d.readLine();
while(true)
{
 out.println(s);
 s=in.readLine();
 System.out.println("Server:>"+s);
 if(s.equalsIgnoreCase("Chatisclosing"))
 break;
 System.out.println("Message:");
 s=d.readLine();
 if(s.equalsIgnoreCase("end"))
 break;
}
}
finally
{
 soc.close();
```

```
  }
 }
}
```

SOURCE CODE FOR CHAT SERVER:

```
import java.io.*;
import java.net.*;

class chatserver
{
 public static void main(String args[])throws IOException
 {
  ServerSocket s=new ServerSocket(0202); try
  {
   while(true)
   {
    Socket soc=s.accept();
    try
    {
     new chat(soc
}
    catch(IOException e)
    {
     soc.close();
    }
```

```
    }

   }
finally
    {
     s.close();
    }
   }
  }
```

SAMPLE INPUT/OUTPUT:

E:\EX8>javac chatserver.java

E:\EX8>java chatserver

Message :

Hi Client, how are you? BE IT:>

I am fine.

E:\EX8>javac chatclient.java

E:\EX8>java chatclient Chat

sessions begins.. Server : Your

name please : BE IT

Server:> Hi Client, howare you?

Message :

I am fine.

Ex. No. 9.

Date:

# Implementation of multicast

Aim

To write a java program for multicast implementation.

Theory

Multicast

Multicast is communication between a single sender and multiple receivers on a network. Typical uses include the updating of mobile personnel from a home office and the periodic issuance of online newsletters. Together with anycast and unicast, multicast is one of the packet types in the Internet Protocol Version 6 (IPv6).

MulticastSocket

public MulticastSocket()
throws IOException
Create a multicast socket.

If there is a security manager, its checkListen method is first called with 0 as its argument to ensure the operation is allowed. This could result in a SecurityException.

Throws:
IOException - if an I/O exception occurs while creating the MulticastSocket SecurityException - if a security manager exists and its checkListen method doesn't allow the operation.

public MulticastSocket(int port)
throws IOException
Create a multicast socket and bind it to a specific port.

If there is a security manager, its checkListen method is first called with the port argument as its argument to ensure the operation is allowed. This could result in a SecurityException.

When the socket is created the DatagramSocket.setReuseAddress(boolean) method is called to enable the SO_REUSEADDR socket option.

Parameters:
port - port to user

SOURCE CODE FOR MULTICAST SOURCE:

```
import
java.io.*;
import
java.net.*;

public class msource
{
 public static void main(String args[])
 {
  try
  {
   DatagramSocket s=new
   DatagramSocket(); byte[] smsg=new
   byte[100]; System.out.println("Enter the
   text to send : "); int
   len=System.in.read(smsg);
   InetAddress     test=InetAddress.getLocalHost();
   DatagramPacketpack=new
   DatagramPacket(smsg,len,test,16900); s.send(pack);
   s.close();
  }
 catch(Exception err)
 {
  System.out.println(err);
 }
}
```