2009

DATA STRUCTURES LAB MANUAL

CHESTI ALTAFF HUSSAIN

LECTURER, DEPARTMENT OF ECE

BAPATLA ENGINEERING COLLEGE

BAPATLA

BAPATLA ENGINEERING COLLEGE, BAPATLA
DEPARTMENT OF ELECTRONICS AND COMMUNICATIONS ENGINEERING
EC 263: DATA STRUCTURES LAB MANUAL

Prepared by

CHESTI ALTAFF HUSSAIN
M.Tech
Lecturer
Department of ECE

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)
 {
 case 1: fp=p=p1=(struct student *)malloc(sizeof(struct student));
```

```
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);
}
}
```