/*
Program 1) Write a C program to create a sequential file with atleast five records, each record having the structure shown below:
* USN (Non-zero positive integer)
* Name (25 Characters)
* Marks1 (Positive integer)
* Marks2 (Positive integer)
* Marks3 (Positive integer)
Write necessary functions 1) To display all the records in the file.
2) To search for a specific record based on the USN. In case the required record is not found, suitable message should be displayed.Both the options in this case must be demonstrated.
*/

```
#include<stdio.h>
#include<process.h>
#include<string.h>
struct student
{
unsigned usn;
char name[25];
unsigned marks1;
unsigned marks2;
unsigned marks3;
};

typedef struct student stud;
stud s;

int search_record(int key,FILE *fp)
{
while(1)
{
fread(&s,sizeof(s),1,fp);
if(feof(fp)) break;
if(key==s.usn)   return 1;
}
return 0;
}

void insert_record(FILE *fp)
{
unsigned usn;
int n;
printf("Enter a USN :-");
scanf("%u",&usn);
```

```
if(usn==0)
{
printf("\n Invalid USN ...Retry\n");
return;
}
n=search_record(usn,fp);
if(n==1)
{
printf("\nDuplicate USN !! Retry\n");
return;
}
s.usn=usn;
printf("Name :- ");
scanf("%s",s.name);
printf("Marks 1 :- ");
scanf("%u",&s.marks1);
printf("Marks 2 :- ");
scanf("%u",&s.marks2);
printf("Marks 3 :- ");
scanf("%u",&s.marks3);
fwrite(&s,sizeof(s),1,fp);
}

void display_record(FILE *fp)
{
printf("\nUSN  Name   Marks1   Marks2   Marks3\n");
while(1)
{
fread(&s,sizeof(s),1,fp);
if(feof(fp))  break;
printf("%-5u%-12s%-5u\t%-5u\t%-5u\n",s.usn,s.name,s.marks1,s.marks2,s.marks3);
}
}

void main()
{
char fname[10];
FILE *fp=NULL;
int i,key,choice,n;
clrscr();
while(1)
{
printf("1.Insert record\n2. Search record\n");
printf("3. Display record\n4. Quit\n");
printf("Enter your  choice :-");
scanf("%d",&choice);
```

```
switch(choice)
{
case 1: fp=fopen("STD.DAT","a+");
if(fp==NULL)
{
printf("Sorry! Unable to open File \n");
break;
}
insert_record(fp);
fclose(fp);
break;
case 2: fp=fopen("STD.DAT","r+");
if(fp==NULL)
{
printf("Sorry! Unable to open File \nor 0 records present\n");
break;
}
printf("\nEnter USN to be searched :");
scanf("%d",&key);
n=search_record(key,fp);
if(n==1)
{
printf("\nRecords found...Details are...\n");
printf("\nUSN  Name  Marks1  Marks2  Marks3\n");
printf("%-5u%-12s%-5u\t%-5u\t%-5u\n",s.usn,s.name,s.marks1,s.marks2,s.marks3);
fclose(fp);
break;
}
printf("\nRecord not found in file\n");
fclose(fp);
break;
case 3: fp=fopen("STD.DAT","r+");
if(fp==NULL)
{
printf("Sorry Unable to open file \n or 0 records present\n");
break;
}
display_record(fp);
fclose(fp);break;
default :getch();
        return;
}
}
}
```

Output :-

```
1.Insert record
2. Search record
3. Display record
4. Quit
Enter your  choice :-1
Enter a USN :-1
Name :- Vijay
Marks 1 :- 25
Marks 2 :- 22
Marks 3 :- 24
1.Insert record
2. Search record
3. Display record
4. Quit
Enter your  choice :-1
Enter a USN :-2
Name :- Hemant
Marks 1 :- 15
Marks 2 :- 18
Marks 3 :- 24
1.Insert record
2. Search record
3. Display record
4. Quit
Enter your  choice :-1
Enter a USN :-3
Name :- Vinayak
Marks 1 :- 20
Marks 2 :- 21
Marks 3 :- 23
1.Insert record
2. Search record
3. Display record
4. Quit
Enter your  choice :- 1
Enter a USN :-4
Name :- Rajkumar
Marks 1 :- 20
Marks 2 :- 21
Marks 3 :- 23
1.Insert record
2. Search record
3. Display record
```

4. Quit
Enter your choice :-1
Enter a USN :-5
Name :- Antosh
Marks 1 :- 21
Marks 2 :- 20
Marks 3 :- 18
1.Insert record
2. Search record
3. Display record
4. Quit
Enter your choice :-2

Enter USN to be searched :1

Records found...Details are...

USN	Name	Marks1	Marks2	Marks3
1	Vijay	25	22	24

1.Insert record
2. Search record
3. Display record
4. Quit
Enter your choice :-3

USN	Name	Marks1	Marks2	Marks3
1	Vijay	25	22	24
2	Hemant	15	18	24
3	Vinayak	20	21	23
4	Rajkumar	20	21	23
5	Antosh	21	20	18

1.Insert record
2. Search record
3. Display record
4. Quit
Enter your choice :-1
Enter a USN :-2

Duplicate USN !! Retry
1.Insert record
2. Search record
3. Display record
4. Quit
Enter your choice :- 4

```
/*
 Program 2) Write a C program to construct a stack of integers and to perform the following operations on it.
        a) Push
        b) Pop
        c) Display
*/
#include<stdio.h>
#include<conio.h>
#define max 10
int stack[max],ele,i,choice,top=-1,rpt;

void push(int ele)
{
if(top>=(max-1))
{
printf("Stack is over flow\n");
return;
}
stack[++top]=ele;
}

void pop()
{
if(top==-1)
{
printf("Stack is under flow\n");
return;
}
printf("Deleted element is %d\n",stack[top--]);
}

void display()
{
if(top==-1)
{
printf("Stack is underflow\n\n");
return;
}
printf("Stack contents are\n");
for(i=top;i>=0;i--)
{
printf("%d\n",stack[i]);
}
return;
}
```

```
void main()
{
clrscr();
while(1)
{
printf("Stack operations\n");
printf("1. Push\n");
printf("2. Pop\n");
printf("3. Display\n");
printf("4. Exit\n");
printf("Enter your choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1: printf("Enter the element to be pushed\n");
        scanf("%d",&ele);
        push(ele);
        break;
case 2: pop();
        break;
case 3: display();
        break;
case 4 : getch();
        return;
default : printf("Invalid Choice\n");
}
}

}
```

Output :-

```
Stack operations
1. Push
2. Pop
3. Display
4. Exit
Enter your choice
1
Enter the element to be pushed
10
Stack operations
1. Push
2. Pop
3. Display
4. Exit
Enter your choice
1
Enter the element to be pushed
20
Stack operations
1. Push
2. Pop
3. Display
4. Exit
Enter your choice
1
Enter the element to be pushed
30
Stack operations
1. Push
2. Pop
3. Display
4. Exit
Enter your choice
3
Stack contents are
30
20
10
Stack operations
1. Push
2. Pop
3. Display
4. Exit
Enter your choice
```

2
Deleted element is 30
Stack operations
1. Push
2. Pop
3. Display
4. Exit
Enter your choice
3
Stack contents are
20
10
Stack operations
1. Push
2. Pop
3. Display
4. Exit
Enter your choice 4

/*
Program 3) Write a C program to convert and print a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide)
*/

```
#include<stdio.h>
#include<conio.h>
char stack[10],infix[10],postfix[10],temp;
int top=-1,i,j=0;
void push(char x)
{
stack[++top]=x;
return;
}
char pop()
{
return stack[top--];
}
int prior(char x)
{
int p;
if (x=='('||x=='#')p=1;
if(x=='+'||x=='-')p=2;
if(x=='*'||x=='/')p=3;
if(x=='$'||x=='^')p=4;
return p;
}



void main()
{
clrscr();
printf("Enter a valid infix expression\n");
scanf("%s",infix);
push('#');
for(i=0;i<strlen(infix);i++)
{
if (infix[i]=='(')
push(infix[i]);
else if(infix[i]==')')
{
temp=pop();
while(temp!='(')
```

```
{
postfix[j++]=temp;
temp=pop();
}
}
else if(isalnum(infix[i]))
postfix[j++]=infix[i];
else
{
while(prior(stack[top])>=prior(infix[i]))
{
temp=pop();
postfix[j++]=temp;
}
push(infix[i]);
}
}
while(stack[top]!='#')
{
temp=pop();
postfix[j++]=temp;
}
postfix[j]='\0';
printf("Postfix expression is %s\n",postfix);
getch();
}
```

Output :-

```
Enter a valid infix expression
a*b+c
Postfix expression is ab*c+

Enter a valid infix expression
a+b*c
Postfix expression is abc*+

Enter a valid infix expression
a+(b*c)+d
Postfix expression is abc*+d+
```

/*
Program 4) Write a C program to evaluate a valid suffix/postfix expression using stack. Assume that the suffix / postfix expression is read as a single line consisting of non-negative single digit operands and binary operator.
The arithmetic operators are + (ADD), - (SUBTRACT), * (MULTIPLY) and / (DIVIDE).
*/

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
float stack[20],x,y,res;
char s[10];
int i,top=-1;
void push(float x)
{
stack[++top]=x;
return;
}
float pop()
{
return stack[top--];
}
void main()
{
clrscr();
printf("Enter a valid suffix expression\n");
gets(s);
for(i=0;i<strlen(s);i++)
{
if(isdigit(s[i]))
push(s[i]-'0');
else
{
y=pop();
x=pop();
switch(s[i])
{
case '+':push(x+y);
        break;
case '-':push(x-y);
        break;
case '*':push(x*y);
        break;
case '/':push(x/y);
        break;
```

```
case '^':push(pow(x,y));
            break;
}
}
}
res=pop();
printf("Result of suffix expression %s =%f\n",s,res);
getch();
}
```

Output :-

```
Enter a valid suffix expression
576*+
Result of suffix expression 576*+ =47.000000

Enter a valid suffix expression
945*-
Result of suffix expression 945*- =-11.000000

Enter a valid suffix expression
53^
Result of suffix expression 53^ =125.000000
```

/*
Program 5) Write a C program to simulate the working of a queue of integers using an array. Provide the following operations.

a) Insert
b) Delete
c) Display

*/

```
#include<stdio.h>
#include<conio.h>
#define MAX 10
int q[10],front=-1,rear=-1,ele,i,ch;

qinsert(int ele)
{
if(rear>=(MAX-1))
{
printf("Queue is full\n");
return;
}
q[++rear]=ele;
if(front==-1)
front++;
return;
}

qdelete()
{
if(front==-1)
{
printf("Queue is empty\n");
return;
}
printf("Deleted element is %d\n",q[front]);
if(front==rear)
front=rear=-1;
else
front=front+1;
return;
}

qdisplay()
{
if(front==-1)
{
printf("Queue is empty\n");
```

```
return;
}
printf("Status of Queue is\n");
for(i=front;i<=rear;i++)
printf("%d\t",q[i]);
return;
}

void main()
{
clrscr();
while(1)
{
printf("Queue operations\n");
printf("1. Q Insert\n");
printf("2. Q Delete\n");
printf("3. Q Display\n");
printf("Enter your choice\n");
scanf("%d",&ch);
switch(ch)
{
case 1:printf("Enter the element to be inserted\n");
     scanf("%d",&ele);
     qinsert(ele);
     break;
case 2:qdelete();
      break;
case 3:qdisplay();
     break;
default: getch();
           return;
}
}
}
```

Output :-

```
Queue operations
1. Q Insert
2. Q Delete
3. Q Display
Enter your choice
1
Enter the element to be inserted
10
Queue operations
1. Q Insert
```

```
2. Q Delete
3. Q Display
Enter your choice
1
Enter the element to be inserted
20
Queue operations
1. Q Insert
2. Q Delete
3. Q Display
Enter your choice
1
Enter the element to be inserted
30
Queue operations
1. Q Insert
2. Q Delete
3. Q Display
Enter your choice
3
Status of Queue is
10      20      30
Queue operations
1. Q Insert
2. Q Delete
3. Q Display
Enter your choice
2
Deleted element is 10
Queue operations
1. Q Insert
2. Q Delete
3. Q Display
Enter your choice
3
Status of Queue is
20      30
Queue operations
1. Q Insert
2. Q Delete
3. Q Display
Enter your choice
4
```

```
/*
Program 6) Write a C program to simulate the working of a circular queue of integers
using an array. Provide the following operations
a) Insert
b) Delete
c) Display
*/

#include<stdio.h>
#include<conio.h>
#define max 3
int cq[10],front=-1,rear=-1;
int ele,i,ch;
cqinsert(int ele)
{
if(front==(rear+1)%max)
{
printf("Circular Queue is full\n");
return;
}
rear=(rear+1)%max;
cq[rear]=ele;
if(front==-1)
front++;
return;
}

cqdelete()
{
if(front==-1)
{
printf("Circular Queue is empty\n");
return;
}
printf("Deleted element is %d\n",cq[front]);
if(front==rear)
front=rear=-1;
else
front=(front+1)%max;
return;
}

cqdisplay()
{
if(front==-1)
{
```

```
printf("Circular Queue is empty\n");
return;
}
else
{
printf(" Status of Circular Queue is\n");
for(i=front;i<=rear;i++)
printf("%d\n",cq[i]);
}
if(front>rear)
{
for(i=front;i<max;i++)
printf("%d\n",cq[i]);
for(i=0;i<=rear;i++)
printf("%d\n",cq[i]);
}
return;
}

void main()
{
clrscr();
while(1)
{
printf(" Circular Queue Operations\n");
printf("1. Circular Queue Insert\n");
printf("2. Circular Queue Delete\n");
printf("3. Circular Queue Display\n");
printf(" Enter your choice\n");
scanf("%d",&ch);
switch (ch)
{
case 1:printf(" Enter the element to be inserted\n");
      scanf("%d",&ele);
      cqinsert(ele);
      break;
case 2: cqdelete();
      break;
case 3: cqdisplay();
      break;
default: getch();
        return;
}
}
}
```

Output :-

```
Circular Queue Operations
1. Circular Queue Insert
2. Circular Queue Delete
3. Circular Queue Display
 Enter your choice
1
 Enter the element to be inserted
10
 Circular Queue Operations
1. Circular Queue Insert
2. Circular Queue Delete
3. Circular Queue Display
 Enter your choice
1
 Enter the element to be inserted
20
 Circular Queue Operations
1. Circular Queue Insert
2. Circular Queue Delete
3. Circular Queue Display
 Enter your choice
1
 Enter the element to be inserted
30
 Circular Queue Operations
1. Circular Queue Insert
2. Circular Queue Delete
3. Circular Queue Display
 Enter your choice
3
 Status of Circular Queue is
10
20
30
 Circular Queue Operations
1. Circular Queue Insert
2. Circular Queue Delete
3. Circular Queue Display
 Enter your choice
2
Deleted element is 10
 Circular Queue Operations
1. Circular Queue Insert
2. Circular Queue Delete
```

3. Circular Queue Display
Enter your choice
3
Status of Circular Queue is
20
30
Circular Queue Operations
1. Circular Queue Insert
2. Circular Queue Delete
3. Circular Queue Display
Enter your choice
4

```
/*
Program 7) Write a program to design a Priority Queue
 which is maintained as a set of queues (assume a maximum  of 3 queues).
 The elements are inserted based  upon the given priority.
 The deletion of an element is to be done starting from the 1st queue,
 if it is not empty. If it is empty, the elements from 2nd queue  will
 be deleted and so on.
*/
#include<stdio.h>
#include<conio.h>
#define max 5
int ch;
struct queue
{
int q[max];
int front;
int rear;
}pq[4];

void pqinsert()
{
int i,ele;
printf("Enter the element to be inserted\n");
scanf("%d",&ele);
printf("Enter Priority 1,2 or 3\n");
scanf("%d",&i);
if(i>3)
{
printf("\nInvalid Priority. Retry\n");
return;
}
if(pq[i].rear==(max-1))
{
printf("Queue %d is FULL. Try to insert on to next Priority Queue\n",i);
return;
}
pq[i].rear++;
pq[i].q[pq[i].rear]=ele;
if(pq[i].front==-1)
pq[i].front=0;
}

void pqdelete()
{
int i=0,flag=0;
for(i=1;i<=3;i++)
```

```
if(pq[i].front==-1)
printf("\n Queue  %d: EMPTY",i);
else if(pq[i].front!= -1)
{
printf("\n Element is deleted from Queue %d\n",i);
flag=1;
break;
}
if(flag==0) return;
printf("Deleted Element is %d\n",pq[i].q[pq[i].front]);
if(pq[i].front==pq[i].rear)
pq[i].front=pq[i].rear=-1;
else
pq[i].front++;
}

void pqdisplay()
{
int i,j;
printf("Queue Status :-");
for(i=1;i<=3;i++)
{
printf("\nQueue %d:- ",i);
if(pq[i].front==-1)
printf("EMPTY");
else
for(j=pq[i].front;j<=pq[i].rear;j++)
printf("%d\t",pq[i].q[j]);
}
}

void main()
{
int i;
clrscr();
for(i=1;i<=3;i++)
{
pq[i].front=pq[i].rear=-1;
}
while(1)
 {
 printf("\nPriority Queue Operations\n");
 printf("1. Priority Queue Insert\n");
 printf("2. Priority Queue Delete\n");
 printf("3. Priority Queue Display\n");
 printf("Enter your choice\n");
```

```
 scanf("%d",&ch);
 switch(ch)
 {
  case 1:  pqinsert();
  break;
  case 2:  pqdelete();
  break;
  case 3:  pqdisplay();
  break;
  default: getch();
           return;
  }
 }
}
```

Output :-

```
Priority Queue Operations
1. Priority Queue Insert
2. Priority Queue Delete
3. Priority Queue Display
Enter your choice
1
Enter the element to be inserted
10
Enter Priority 1,2 or 3
1

Priority Queue Operations
1. Priority Queue Insert
2. Priority Queue Delete
3. Priority Queue Display
Enter your choice
1
Enter the element to be inserted
20
Enter Priority 1,2 or 3
1

Priority Queue Operations
1. Priority Queue Insert
2. Priority Queue Delete
3. Priority Queue Display
Enter your choice
1
Enter the element to be inserted
```

```
30
Enter Priority 1,2 or 3
1

Priority Queue Operations
1. Priority Queue Insert
2. Priority Queue Delete
3. Priority Queue Display
Enter your choice
3
Queue Status :-
Queue 1:- 10    20      30
Queue 2:- EMPTY
Queue 3:- EMPTY
Priority Queue Operations
1. Priority Queue Insert
2. Priority Queue Delete
3. Priority Queue Display
Enter your choice
1
Enter the element to be inserted
15
Enter Priority 1,2 or 3
2

Priority Queue Operations
1. Priority Queue Insert
2. Priority Queue Delete
3. Priority Queue Display
Enter your choice
1
Enter the element to be inserted
25
Enter Priority 1,2 or 3
3

Priority Queue Operations
1. Priority Queue Insert
2. Priority Queue Delete
3. Priority Queue Display
Enter your choice
3
Queue Status :-
Queue 1:- 10    20      30
Queue 2:- 15
Queue 3:- 25
```

Priority Queue Operations
1. Priority Queue Insert
2. Priority Queue Delete
3. Priority Queue Display
Enter your choice
2

Element is deleted from Queue 1
Deleted Element is 10

Priority Queue Operations
1. Priority Queue Insert
2. Priority Queue Delete
3. Priority Queue Display
Enter your choice
3
Queue Status :-
Queue 1:- 20 30
Queue 2:- 15
Queue 3:- 25
Priority Queue Operations
1. Priority Queue Insert
2. Priority Queue Delete
3. Priority Queue Display
Enter your choice

/*
Program 8) Write a C program using dynamic variables & pointers, to construct a singly linked list consisting of the following information in each node :
student id(integer), student name (character string) & semester (integer)
The operations to be supported are :
a) The insert operation
I) At the front of a list
II) At the back of the list
III) At any position in the list.
b) deleting a node based on student id.
c) Searching a node based on Id & update the information content.
d) Displaying all the nodes in the list.
*/

```
#include<stdio.h>
#include<string.h>
#include<alloc.h>
int sid,sem,ch,choice,pos,count;
char name[20];
typedef struct node
{
  int sid;
  int sem;
  char name[20];
  struct node *link;
}NODE;
NODE *start=NULL;

void front_insert()
{
 NODE *nn;
 nn=(NODE*)malloc(sizeof(NODE));
 nn->link=NULL;
 printf(" Enter SID\n");
 scanf("%d",&sid);
fflush(stdin);
 printf(" Enter Name \n");
 scanf("%s",name);
 printf(" Enter Semester\n");
 scanf("%d",&sem);
 nn->sid=sid;
 nn->sem=sem;
 strcpy(nn->name,name);
 nn->link=start;
 start=nn;
}
```

```
void end_insert()
{
NODE *nn,*cur;
nn=(NODE*)malloc(sizeof(NODE));
printf(" Enter SID\n");
 scanf("%d",&sid);
fflush(stdin);
 printf(" Enter Name \n");
 scanf("%s",name);
 printf(" Enter Semester\n");
 scanf("%d",&sem);
nn->sid=sid;
nn->sem=sem;
strcpy(nn->name,name);
nn->link=NULL;
if(start==NULL)
{
start=nn;
return;
}
cur=start;
while(cur->link!=NULL)
 cur=cur->link;
cur->link=nn;
return;
}

void insert_pos()
{
  NODE *nn,*prev,*cur;
  nn=(NODE*)malloc(sizeof(NODE));
 printf(" Enter SID\n");
 scanf("%d",&sid);
fflush(stdin);
 printf(" Enter Name \n");
 scanf("%s",name);
 printf(" Enter Semester\n");
 scanf("%d",&sem);
  nn->sid=sid;
  nn->sem=sem;
  strcpy(nn->name,name);
  nn->link=NULL;
  printf("enter the position\n");
  scanf("%d",&pos);
  if(start==NULL&&pos==1)
```

```
  {
    start=nn;
    return;
  }
  if(pos==1)
  {
   nn->link=start;
   start=nn;
   return;
  }
  count=1;
  prev=start;
  cur=start->link;
  while(cur!=NULL)
  {
    count++;
    if(count==pos)
    {
      prev->link=nn;
      nn->link=cur;
      return;
    }
    prev=cur;
    cur=cur->link;
  }
  printf("Invalid position\n");
  return;
}

 void del()
 {
  NODE *prev,*cur;
  if(start==NULL)
  {
   printf("List is empty.Deletion is not possible\n");
   return;
  }
  printf("Enter the sid to be deleted\n");
  scanf("%d",&sid);
  if(start->sid==sid)
  {
    printf("Deleted node is%d\n",start->sid);
    start=start->link;
    return;
  }
  prev=start;
```

```
  cur=start->link;
  while(cur!=NULL)
  {
    if(cur->sid==sid)
    {
      printf("Deleted node is%d\n",cur->sid);
      prev->link=cur->link;
      return;
    }
    prev=prev->link;
    cur=cur->link;
  }
}

  search()
  {
  NODE *cur;
  if(start==NULL)
  {
   printf(" List is empty\n");
   return;
  }
  printf("Enter the sid to be searched\n");
  scanf("%d",&sid);
  cur=start;
  while(cur!=NULL)
  {
   if(cur->sid==sid)
   {
    printf("ID%d is found in the list\n",sid);
    printf("sid:%d\n name:%s\n sem:%d\n",cur->sid,cur->name,cur->sem);
    printf("Do you want to up date?1:0\n");
    scanf("%d",&choice);
    if(choice==1)
    {

  printf(" Enter SID\n");
  scanf("%d",&cur->sid);
  fflush(stdin);
  printf(" Enter Name \n");
  scanf("%s",cur->name);
  printf(" Enter Semester\n");
  scanf("%d",&cur->sem);
  printf("Modified Information is\n");
  printf("sid:%d\nname:%s\nsem:%d\n",cur->sid,cur->name,cur->sem);
    }
```

```
 return;
 }
 cur=cur->link;
}
 printf("Student with id%not found in the list\n",sid);
 return;
}

display()
{
NODE *cur;
if(start==NULL)
{
 printf("list is empty\n");
 return;
}
cur=start;
while(cur!=NULL)
{
printf("|%d|->|%s|->|%d|->\n",cur->sid,cur->name,cur->sem);
cur=cur->link;
}
printf("NULL\n");
return;
}

main()
{
clrscr();
while(1)
{
 printf("Operations of linked list\n");
 printf("1.Front insert\n");
 printf("2.End insert\n");
 printf("3.Insert at specified position\n");
 printf("4.Delete\n");
 printf("5.Search\n");
 printf("6.Display\n");
 printf("Enter your choice\n");
 scanf("%d",&ch);
 switch(ch)
 {
 case 1:front_insert();
          break;
 case 2:end_insert();
          break;
```

```
case 3: insert_pos();
                break;
case 4:del();
        break;
case 5:search();
        break;
case 6:display();
        break;
default:getch();
      return;
}
}
}
```

Output:-

```
Operations of linked list
1.Front insert
2.End insert
3.Insert at specified position
4.Delete
5.Search
6.Display
Enter your choice
1
 Enter SID
1
 Enter Name
Vijay
 Enter Semester
4
Operations of linked list
1.Front insert
2.End insert
3.Insert at specified position
4.Delete
5.Search
6.Display
Enter your choice
1
 Enter SID
2
 Enter Name
Patil
 Enter Semester
```

```
3
Operations of linked list
1.Front insert
2.End insert
3.Insert at specified position
4.Delete
5.Search
6.Display
Enter your choice
6
|2|->|Patil|->|3|->
|1|->|Vijay|->|4|->
NULL
Operations of linked list
1.Front insert
2.End insert
3.Insert at specified position
4.Delete
5.Search
6.Display
Enter your choice
2
Enter SID
3
Enter Name
Vinayak
 Enter Semester
5
Operations of linked list
1.Front insert
2.End insert
3.Insert at specified position
4.Delete
5.Search
6.Display
Enter your choice
6
|2|->|Patil|->|3|->
|1|->|Vijay|->|4|->
|3|->|Vinayak|->|5|->
NULL
Operations of linked list
1.Front insert
2.End insert
3.Insert at specified position
4.Delete
```

5.Search
6.Display
Enter your choice
4
Enter the sid to be deleted
2
Deleted node is2
Operations of linked list
1.Front insert
2.End insert
3.Insert at specified position
4.Delete
5.Search
6.Display
Enter your choice
6
|1|->|Vijay|->|4|->
|3|->|Vinayak|->|5|->
NULL
Operations of linked list
1.Front insert
2.End insert
3.Insert at specified position
4.Delete
5.Search
6.Display
Enter your choice
7

/*
Program 9) Write a C program using dynamic variables and pointers to construct a stack of integers using singly linked list and to perform following operations
a) Push
b) Pop
c) Display
The program should print appropriate message as per stack overflow & stack empty.
*/

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#define size 5
int count=0;
int ele,ch;
typedef struct node
{
 int data;
 struct node *link;
}NODE;
NODE *top=NULL;

void push()
{
  NODE *nn;
  if(count==size)
  {
    printf("Stack is full, overflow\n");
    return;
  }
  nn=(NODE*)malloc(sizeof(NODE));
  printf("Enter the element to be pushed\n");
  scanf("%d",&ele);
  nn->data=ele;
  nn->link=top;
  top=nn;
  count++;
}

void pop()
{
  if(top==NULL)
  {
    printf("Stack is empty, underflow\n");
    return;
  }
```

```
  printf("Deleted element is%d\n",top->data);
  top=top->link;
  count--;
}

void display()
{
  NODE *temp;
  if(top==NULL)
  {
    printf("Stack is empty, underflow\n");
    return;
  }
  temp=top;
  printf("Content of stack are\n");
  while(temp!=NULL)
  {
    printf("%d\n",temp->data);
    temp=temp->link;
  }
}

void main()
{
 clrscr();
 while(1)
 {
   printf("Stack operations\n");
   printf("1. Push\n");
   printf("2. Pop\n");
   printf("3. Display\n");
   printf("Enter your choice\n");
   scanf("%d",&ch);
   switch(ch)
   {
     case 1:push();
                break;
     case 2:pop();
                break;
     case 3:display();
                break;
     default:getch();
               return;
     }
   }
}
```

Output :-

```
Stack operations
1. Push
2. Pop
3. Display
Enter your choice
1
Enter the element to be pushed
10
Stack operations
1. Push
2. Pop
3. Display
Enter your choice
1
Enter the element to be pushed
20
Stack operations
1. Push
2. Pop
3. Display
Enter your choice
3
Content of stack are
20
10
Stack operations
1. Push
2. Pop
3. Display
Enter your choice
2
Deleted element is20
Stack operations
1. Push
2. Pop
3. Display
Enter your choice
3
Content of stack are
10
Stack operations
1. Push
2. Pop
3. Display
Enter your choice 4
```

/*
Program 10) Write a C program using dynamic variables and pointers to construct a queue of integers using singly linked list and to perform the following operations.
a) Insert
b) Delete
c) Display
The program should print appropriate messages for the queue full & queue empty
*/

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#define size 5
int count=0;
int ele,ch;
typedef struct node
{
  int data;
  struct node *link;
}NODE;
NODE *front=NULL;
NODE *rear=NULL;

void insert()
{
  NODE *nn;
  if(count==size)
  {
    printf("Queue is full,overflow\n");
    return;
  }
  nn=(NODE*)malloc(sizeof(NODE));
  printf("Enter the element to be inserted\n");
  scanf("%d",&ele);
  nn->data=ele;
  nn->link=NULL;
  if(front==NULL)
  {
    front=rear=nn;
    count++;
    return;
  }
  rear->link=nn;
  rear=nn;
  count++;
}
```

```
void del()
{
 if(front==NULL)
 {
  printf("Queue is empty, underflow\n");
  return;
 }
 printf("Deleted element is%d\n",front->data);
 front=front->link;
 count--;
}

void display()
{
 NODE *temp;
 if(front==NULL)
 {
  printf("Queue is empty, underflow\n");
  return;
 }
 temp=front;
 printf("Content of Queue are\n");
 while(temp!=NULL)
 {
   printf("%d\t",temp->data);
   temp=temp->link;
 }
}


void main()
{
 clrscr();
 while(1)
 {
  printf("\nImplementation of Queue using linked list\n");
  printf("1.Insert\n");
  printf("2.Delete\n");
  printf("3.Display\n");
  printf("Enter your choice\n");
  scanf("%d",&ch);
  switch(ch)
  {
    case 1:insert();
           break;
```

```
    case 2:del();
              break;
    case 3:display();
              break;
    default: getch();
             return;
   }
  }
}
```

Output :-

```
Implementation of Queue using linked list
1.Insert
2.Delete
3.Display
Enter your choice
1
Enter the element to be inserted
10
Implementation of Queue using linked list
1.Insert
2.Delete
3.Display
Enter your choice
1
Enter the element to be inserted
20
Implementation of Queue using linked list
1.Insert
2.Delete
3.Display
Enter your choice
3
Content of Queue are
10      20
Implementation of Queue using linked list
1.Insert
2.Delete
3.Display
Enter your choice
2
Deleted element is10
Implementation of Queue using linked list
1.Insert
2.Delete
```

3.Display
Enter your choice
3
Content of Queue are
20
Implementation of Queue using linked list
1.Insert
2.Delete
3.Display
Enter your choice
4

/*
Program 11) Write a C program to support the following operations on a doubly linked list where each node is consists of integers
a) Create a doubly linked list by adding each node at the front
b) Insert a new node to the left of the node whose key value is read as an input.
c) Delete athe node of given data, if it is found, otherwise display appropriate message
d) Display the contents of the list
*/

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
int ele,ch,key;

typedef struct node
{
  int data;
  struct node *left;
  struct node *right;
}NODE;
NODE *start=NULL;

void addfront()
{
   NODE *nn;
   printf("Enter the element to be inserted\n");
   scanf("%d",&ele);
   nn=(NODE*)malloc(sizeof(NODE));
   nn->data=ele;
   nn->right=NULL;
   nn->left=NULL;
   if(start==NULL)
   {
     start=nn;
     return;
   }
   nn->right=start;
   start->left=nn;
   start=nn;
   return;
}

void insert_before()
{
   NODE *nn,*prev,*cur;
```

```
  if(start==NULL)
  {
    printf("List is empty\n");
    return;
  }
  printf("Enter the key value\n");
  scanf("%d",&key);
  if(start->data==key)
  {
    addfront();
    return;
  }
  prev=start;
  cur=prev->right;
  while(cur!=NULL)
  {
     if(cur->data==key)
     {
          printf("Enter the element to be inserted\n");
          scanf("%d",&ele);
          nn=(NODE*)malloc(sizeof(NODE));
          nn->data=ele;
          nn->right=cur;
          nn->left=prev;
          cur->left=nn;
          prev->right=nn;
          return;
     }
     prev=prev->right;
     cur=cur->right;
  }
  printf("Key value not found\n");
  return;
 }

void del()
 {
  NODE *prev,*cur;
  if(start==NULL)
  {
    printf("List is empty\n");
    return;
  }
  printf("Enter the node element to be deleted\n");
  scanf("%d",&ele);
  if(start->data==ele)
```

```
  {
    printf("Deleted element is%d\n",start->data);
    start=start->right;
    return;
  }
  prev=start;
  cur=prev->right;
  while(cur!=NULL)
  {
    if(cur->data==ele)
    {
      printf("Deleted element is%d\n",cur->data);
      prev->right=cur->right;
      cur->right->left=prev;
      return;
    }
    prev=prev->right;
    cur=cur->right;
  }
  printf("Element is not found in the list\n");
  return;
}


void display()
{
  NODE *cur;
  if(start==NULL)
  {
    printf("List is empty\n");
    return;
  }
  cur=start;
  printf("Status of list is\n");
  while(cur!=NULL)
  {
    printf("%d<->",cur->data);
    cur=cur->right;
  }
    printf("NULL\n");
    return;
  }

void main()
 {
    clrscr();
```

```
  while(1)
  {
   printf("Operations of Doubly Linked List\n");
   printf("1. Add at Front\n");
   printf("2. Insert Before\n");
   printf("3. Delete\n");
   printf("4. Display\n");
   printf("Enter your choice");
   scanf("%d",&ch);
   switch(ch)
   {
        case 1: addfront();
                break;
        case 2: insert_before();
                break;
        case 3: del();
                break;
        case 4: display();
                break;
        default: getch();
            return;
   }
  }
}
```

Output:-

```
Operations of Doubly Linked List
1. Add at Front
2. Insert Before
3. Delete
4. Display
Enter your choice1
Enter the element to be inserted
20
Operations of Doubly Linked List
1. Add at Front
2. Insert Before
3. Delete
4. Display
Enter your choice1
Enter the element to be inserted
10
Operations of Doubly Linked List
1. Add at Front
2. Insert Before
```

```
3. Delete
4. Display
Enter your choice
2
Enter the key value
10
Enter the element to be inserted
15
Operations of Doubly Linked List
1. Add at Front
2. Insert Before
3. Delete
4. Display
Enter your choice4
Status of list is
15<->10<->20<->NULL
Operations of Doubly Linked List
1. Add at Front
2. Insert Before
3. Delete
4. Display
Enter your choice
3
Enter the node element to be deleted
10
Deleted element is10
Operations of Doubly Linked List
1. Add at Front
2. Insert Before
3. Delete
4. Display
Enter your choice4
Status of list is
15<->20<->NULL
Operations of Doubly Linked List
1. Add at Front
2. Insert Before
3. Delete
4. Display
Enter your choice
5
```

```
/*
Program 12)  Write a C program
a) To construct a Binary Search Tree of integers.
b) To travesals the tree using all the methods i.e. Inorder, Preorder and Postorder
c) To display the elements in the tree
 */

#include<stdio.h>
#include<conio.h>
#include<alloc.h>
int ele,ch;
typedef struct node
{
  int data;
  struct node *left;
  struct node *right;
}BTNODE;
BTNODE *root=NULL;

void create(int ele)
{
  BTNODE *nn,*prev,*cur;
  nn=(BTNODE*)malloc(sizeof(BTNODE));
  nn->data=ele;
  nn->left=NULL;
  nn->right=NULL;
  if(root==NULL)
  {
    root=nn;
    return;
  }
  prev=NULL;
  cur=root;
  while(cur!=NULL)
  {
    prev=cur;
    if(ele<cur->data)cur=cur->left;
    else if(ele>cur->data)cur=cur->right;
    else
    {
      printf("Duplicate value. Can't be inserted\n");
      return;
    }
  }
  if(ele<prev->data)
  prev->left=nn;
```

```
  else
  prev->right=nn;
}

void inorder(BTNODE *root)
{
  if(root!=NULL)
  {
    inorder(root->left);
    printf("%d\n",root->data);
    inorder(root->right);
   }
   return;
}

void preorder(BTNODE *root)
{
  if(root!=NULL)
  {
    printf("%d\n",root->data);
    preorder(root->left);
    preorder(root->right);
  }
   return;
}


void postorder(BTNODE *root)
 {
   if(root!=NULL)
   {
     postorder(root->left);
     postorder(root->right);
     printf("%d\n",root->data);
   }
   return;
 }


void main()
 {
   clrscr();
   while(1)
   {
     printf("Operations of Binary Search Tree (BST)\n");
     printf("1. Create\n");
```

```
    printf("2. Inorder\n");
    printf("3. Preorder\n");
    printf("4. Postorder\n");
    printf("Enter your choice\n");
    scanf("%d",&ch);
    switch(ch)
    {
      case 1: printf("Enter the element to be inserted\n");
              scanf("%d",&ele);
              create(ele);
              break;
      case 2: inorder(root);
              break;
      case 3: preorder(root);
              break;
      case 4: postorder(root);
              break;
      default: getch();
             return;
    }
  }
}
```

Output :-

```
Operations of Binary Search Tree (BST)
1. Create
2. Inorder
3. Preorder
4. Postorder
Enter your choice
1
Enter the element to be inserted
10
Operations of Binary Search Tree (BST)
1. Create
2. Inorder
3. Preorder
4. Postorder
Enter your choice
1
Enter the element to be inserted
15
Operations of Binary Search Tree (BST)
1. Create
2. Inorder
```

3. Preorder
4. Postorder
Enter your choice
1
Enter the element to be inserted
5
Operations of Binary Search Tree (BST)
1. Create
2. Inorder
3. Preorder
4. Postorder
Enter your choice
2
5
10
15
Operations of Binary Search Tree (BST)
1. Create
2. Inorder
3. Preorder
4. Postorder
Enter your choice
3
10
5
15
Operations of Binary Search Tree (BST)
1. Create
2. Inorder
3. Preorder
4. Postorder
Enter your choice
4
5
15
10
Operations of Binary Search Tree (BST)
1. Create
2. Inorder
3. Preorder
4. Postorder
Enter your choice
5

/*
Program 13 A) Write a recursive C program for Searching an element on a given list of integers using Binary Search Method.
*/

```
#include<stdio.h>
#include<conio.h>
int a[10],n,key,pos,mid,i;

int binary(int key,int a[10],int low,int high)
{
if(low>high) return (0);
if(low<=high)
{
  mid=(low+high)/2;
  if(key==a[mid])
    return (mid);
  else if(key<a[mid])
          return (binary(key,a,low,mid-1));
      else
       return (binary(key,a,mid+1,high));
}
}

void main()
{
clrscr();
printf("Enter the size of an array\n");
scanf("%d",&n);
printf(" Enter the elements of an array in ascending order\n");
for(i=1;i<=n;i++)
scanf("%d",&a[i]);
printf(" Enter the key element to be searched\n");
scanf("%d",&key);
pos=binary(key,a,1,n);
if(pos==0)
printf(" Key is not found in the array\n");
else
printf(" Key is found at %d position",pos);
getch();
}
```

Output :-

```
Enter the size of an array
5
 Enter the elements of an array in ascending order
1 2 3 4 5
 Enter the key element to be searched
2
 Key is found at 2 position

Enter the size of an array
5
 Enter the elements of an array in ascending order
1 2 3 4 5
 Enter the key element to be searched
8
 Key is not found in the array
```

```
/*
Program 13 B) Write a recursive C program for solving the Tower of Honai problem
*/

#include<stdio.h>
#include<conio.h>
int count=0;

void move_disks(int n,char S, char T,char D)
{
if(n>0)
{
move_disks(n-1,S,T,D);
printf("Move disk %d from %c to %c\n",n,S,D);
count++;
move_disks(n-1,T,D,S);
}
}

void main()
{
int n;
clrscr();
printf("Enter number of disks\n");
scanf("%d",&n);
move_disks(n,'S','D','T');
printf(" Total number of moves=%d\n",count);
getch();
}
```

Output:-

```
Enter number of disks
3
Move disk 1 from S to T
Move disk 2 from S to T
Move disk 1 from D to S
Move disk 3 from S to T
Move disk 1 from D to S
Move disk 2 from D to S
Move disk 1 from T to D
 Total number of moves=7
```

```
Enter number of disks
4
Move disk 1 from S to T
Move disk 2 from S to T
Move disk 1 from D to S
Move disk 3 from S to T
Move disk 1 from D to S
Move disk 2 from D to S
Move disk 1 from T to D
Move disk 4 from S to T
Move disk 1 from D to S
Move disk 2 from D to S
Move disk 1 from T to D
Move disk 3 from D to S
Move disk 1 from T to D
Move disk 2 from T to D
Move disk 1 from S to T
 Total number of moves=15
```