Lab File
Data Structures Using C
(CSIT 124)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Submitted to:	Submitted by:
Dr Nisha Pahal	Shaina Mehta
Ast. Professor	A2305219268
CSE Department, ASET	B.tech. C.S.E.
	3CSE-4Y

AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY

AMITY UNIVERSITY UTTAR PRADESH

NOIDA -201301

Exp No	Assignment Category	Code	Name of Experiment	Date of Allotment	Date of Evaluation	Max Marks	Marks Obtained	Faculty Sign
1	Mandatory Experiment		Basic C Programs	21-07-2020	3-11-2020			
2			Arrays	28-07-2020	3-11-2020			
3			Stacks and Queues	11-07-2020	3-11-2020			
4			Linked List	25-08-2020	3-11-2020			
5			Trees	21-09-2020	3-11-2020			
6			Searching and Sorting Algorithm	13-10-2020	3-11-2020			
7			Graphs	27-10-2020	3-11-2020			
	Viva	Viva						

Basic C Programs

Q1) WAP to find the size of int, float, double and char.

Ans)

Code:

```
#include <stdio.h>
int main()
{
int a,b,c,d;
a=sizeof(int);
b=sizeof(float);
c=sizeof(double);
d=sizeof(char);
printf("\n the size of int is = %d",a);
printf("\n the size of float is = %d",b);
printf("\n the size of double is = %d",c);
printf("\n the size of char is = %d",d);
return 0;
}
```

Output:

```
the size of int is = 4
the size of float is = 4
the size of double is = 8
the size of char is = 1

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

Q2) WAP to swap two numbers.

Ans)

Code:

```
#include <stdio.h>
void swap(int *a,int *b);
int main()
```

```
{
int a,b;
printf("enter the first no");
scanf("%d",&a);
printf("enter the second no");
scanf("%d",&b);
swap(&a,&b);
printf("the swapped numbers are %d and %d",a,b);
return 0;
}
void swap(int *a,int *b)
{
int temp;
temp=*a;
*a=*b;
*b=temp;
}
```

Output:

```
enter the first no23
enter the second no34
the swapped numbers are 34 and 23

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

Q3) WAP to check whether a character is a vowel or not.

Ans)

Code:

```
#include <stdio.h>
int  check(char ch);
int main()
{
```

```
char ch;
printf("enter the alphabet");
scanf("%c",&ch);
int b;
b=check(ch);
if(b==1)
{
printf("\n the alphabet is a vowel");
}
else
{
printf("\n the alphabet is a consnant");
}
return 0;
}
int check(char ch)
{
int a=0;
if(ch=='a'|| ch=='A'|| ch=='e' || ch=='E' || ch=='i' || ch=='I'|| ch=='o'|| ch=='O'|| ch=='u'|| ch=='U')
{
a=1;
}
else
{
a=0;
}
return a;
}
```

Output:

```
enter the alphabeta

the alphabet is a vowel

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

Q4) WAP to find the roots of a quadratic equation.

Ans)

Code:

```
#include <stdio.h>

#include <math.h>

float root1(float a,float b,float d);

float root2(float a,float b,float d);

int main()

{

int a,b,c;

printf("enter the coeficient of x^2");

scanf("%d",&a);

printf("enter the coeficient of x");

scanf("%d",&b);

printf("enter the coeficient of x^0");

scanf("%d",&c);

float d,p,e;

p=pow(b,2);

e=4*a*c;

d=p-e;

if(d<0.00)

{

printf("\n error: the roots does not exist");

}

else
{
float r1,r2;
r1=root1(a,b,d);
r2=root2(a,b,d);
printf("the roots of a quadratric equation are %f and %f",r1,r2);
}
return 0;
}
float root1(float a,float b,float d)
{
float c,e,f;
c=sqrt(d);
e=0.00-b;
f=(e+c)/(2*a);
return f;
}
float root2(float a,float b,float d)
{
float c,e,g;
c=sqrt(d);
e=0.00-b;
g=(e-c)/(2*a);
return g;
}
```

Output:

```
enter the coeficient of x^22
enter the coeficient of x6
enter the coeficient of x^04
the roots of a quadratric equation are -1.000000 and -2.000000

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

Q5) WAP to check whether a character is an alphabet or not.

Ans)

Code:

```
#include <stdio.h>
int checkalpha(char ch);
int main()
{
char ch;
printf("enter character");
scanf("%c",&ch);
int d;
d=checkalpha(ch);
if(d==1)
{
printf("\n It is a alphabet");
}
else
{
printf(" \n It is not an alphabet");
}
return 0;
}
int  checkalpha(char ch)
{
int a=0;
if(ch>='A' && ch<='Z')
{
a=1;
}
else if(ch>='a' && ch<='z')
{
a=1;
}
else
{
a=0;
}
return a;
}
```

Output:

```
enter characterH

It is a alphabet

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

Q6) WAP to display Fibonacci sequences.

Ans)

Code:

```
void fibonacci(int n);
int main()
{
int n;
printf("enter the limit of the series");
scanf("%d",&n);
printf("\n The fiboncci series");
fibonacci(n);
return 0;
}
void fibonacci(int n)
{
```

```
int a,b,c;
a=0;
b=1;
c=1;
printf("\n %d",a);
printf("\n %d",b);
int i=0;
while(i<n-2)
{
printf("\n %d",c);
a=b;
b=c;
c=a+b;
i++;
}
}
```

Output:

```
enter the limit of the series5

The fiboncci series
0
1
1
2
3

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

Q7) WAP to display characters from A to Z using loops.

Ans)

Code:

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

```
{
char ch='A';
int i;
printf("\n the alphabets from A ton Z are");
for(i=0;i<26;i++)
{
char c=ch+i;
printf("\n %c",c);
}
return 0;
}
```

Output:

```
the alphabets from A ton Z are
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

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

Q8) WAP to check whether a number is a palindrome or not.

Ans)

Code:

```
#include <stdio.h>
int main()
{
int n,r,d=0,b;
printf("enter a number");
scanf("%d",&n);
r=n;
while(r>0)
{
b=r%10;
d=d*10+b;
r=r/10;
}
if(n==d)
{
printf("it is a palindrome");
}
else
{
printf("it is not a palindrome");
}
return 0;
}
```

Output:

```
enter a number343
it is a palindrome

...Program finished with exit code 0
```

```
Press ENTER to exit console.
```

Q9) WAP to make a simple calculator using switch case.

Ans)

Code:

```
#include <stdio.h>
int main()
{
int a,b,c;
int e;
printf("enter the first  number");
scanf("%d",&a);
printf("enter the second  number");
scanf("%d",&b);
printf("enter the operation");
scanf("%d",&e);
switch(e)
{
case 1:c=a+b;
printf("the sum is %d",c);
break;
case 2:c=a-b;
printf("the difference is %d",c);
break;
case 3:c=a*b;
printf("the product is %d",c);
break;
case 4:c=a/b;
printf("\n the qoutient is %d",c);
int d;
d=a%b;
printf("\n the remainder is %d",d);

break;

default:printf("the invalid entry");

}

return 0;

}
```

Output:

```
enter the first  number2
enter the second  number3
enter the operation1
the sum is 5
```

Q10) WAP to print the reverse the string using recursion.

Ans)

Code:

```
#include <stdio.h>

void reverse(char str[],int si, int ei);

int main()

{

char str[20];

printf("enter the word");

gets(str);

int i;

int s=0;

for(i=0;str[i]!='\0';i++)

{

s++;

}

int si=0;

int ei;

ei=s-1;
reverse(str,si,ei);
printf("the reverse of a string is");
printf(" ");
puts(str);
return 0;
}
void reverse(char str[],int si, int ei)
{
if(si>=ei)
{
return;
}
char temp;
temp=str[si];
str[si]=str[ei];
str[ei]=temp;
reverse(str,si+1,ei-1);
return;
}
```

Output:

```
enter the word shaina
the reverse of a string is aniahs


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

Q11) State and prove whether these statements are true or false.

(a) There are 4 arithmetic operations that are used in pointers.

(b) You can define arrays to hold a number of pointers.

(c) C allows you to have a pointer on a pointer.

(d) Passing an argument by reference or by address enabled the passed arguments to be changed in the calling function .

(e) C allows a function to return a pointer to the local variables.

Ans)

(a) It is true that there are 4 arithmetic operations in pointers i.e. ++, --, +, and –.

Code:

```
#include <stdio.h>
int main()
{
int i=10;
int *j=&i;
printf("\n value %d",i++);
printf("\n value %d",*j++);
printf("\n value %d",i--);
printf("\n value %d",(*j)--);
printf("\n address %p",j++);
printf("\n adderss %p",j--);
printf("\n address %p",j+1);
printf("\n address %p",j-1);
return 0;
}
```

Output:

```
value 10
value 11
value 11
value -62410184
address 0x7ffffc47b237
adderss 0x7ffffc47b23b
address 0x7ffffc47b23b
address 0x7ffffc47b233

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

Note:

We cannot use arithmetic operations on two pointers since pointers are storing memory addresses of the value and we cannot perform arithmetic operations on pointers

For example:

```
int main()
{
int i=10;
int j=11;
int *m=&i;
int *n=&j;
printf("\n %p",m+n);
return 0;
}
```

On compilation this will show an error in the code.

```
main.c: In function ‘main’:
main.c:33:21: error: invalid operands to binary + (have ‘int *’ and ‘int *’)
printf("\n %p",m+n);
^
```

(b) It is true that we can create the array to store the address of the pointers that is by creating array of pointers.

Code:

```
#include <stdio.h>
int main()
{
int i=10;
int j=11;
int k=12;
int l=13;
int *a[4];
a[0]=&i;
a[1]=&j;
a[2]=&k;
a[3]=&l;
printf("\n %p",&i);
printf("\n %p",a[0]);
printf("\n %p",&j);
printf("\n %p",a[1]);
printf("\n %p",&k);
printf("\n %p",a[2]);
printf("\n %p",&l);
printf("\n %p",a[3]);
return 0;
}
```

Output:

```
0x7ffe4f113690
0x7ffe4f113690
0x7ffe4f113694
0x7ffe4f113694
0x7ffe4f113698
0x7ffe4f113698
0x7ffe4f11369c
0x7ffe4f11369c

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

(c) true, we can assign pointer to a pointer which is a double pointer.

Code:

```
#include <stdio.h>
int main()
{
int i=10;
int *j=&i;
int **k=&j;
printf("\n value of i %d",i);
printf("\n value of i %d",*j);
printf("\n value of i %d",**k);
printf("\n address of i %p",&i);

printf("\n address of i %p",j);

printf("\n address of i %p",*k);

printf("\n address of j %p",&j);

printf("\n address of j %p",k);

printf("\n address of k %p",&k);

return 0;

}
```

Output:

```
value of i 10
value of i 10
value of i 10
address of i 0x7ffe9f3ad10c
address of i 0x7ffe9f3ad10c
address of i 0x7ffe9f3ad10c
address of j 0x7ffe9f3ad110
address of j 0x7ffe9f3ad110
address of k 0x7ffe9f3ad118

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

(d) True, it will if the same result if we pass the reference or the address in the function.

Code:

```
#include <stdio.h>

void swap2(int *a,int *b);

void swap2(int *a,int *b);

int main()

{

int i=10;

int j=20;

int *p=&i;

int *q=&j;

swap2(p,q);

int k=10;
int l=20;
printf("\n %d %d",i,j);
swap3(&k,&l);
printf("\n %d %d",k,l);
return 0;
}
void swap2(int *a,int *b)
{
int temp;
temp=*a;
*a=*b;
*b=temp;
}
void swap3(int *a,int *b)
{
int temp;
temp=*a;
*a=*b;
*b=temp;
}
```

Output:

```
main.c:22:5: warning: implicit declaration of function ‘swap3’ [-Wimplicit-function-
declaration]
main.c:33:6: warning: conflicting types for ‘swap3’
main.c:22:5: note: previous implicit declaration of ‘swap3’ was here

20 10
20 10

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

(e) True, we can return a pointer in the function.

Code:

```
#include <stdio.h>
int  f(int *k);
int main()
{
int i=10;
int *j=&i;
int **c=f(j);
printf("\n %p",c);
return 0;
}
int f(int *k)
{
int **c=&k;
return *c;
}
```

Output:

```
main.c:15:13: warning: initialization makes pointer from integer without a cast [-Wint-
conversion]
main.c:22:12: warning: return makes integer from pointer without a cast [-Wint-
conversion]

0xffffffffdb0a08ec

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

Note: This will give correct result in case of double pointer otherwise it will give you segmentation fault or garbage address of a pointer or errors in the code and its execution.

Arrays

Q1) Write a program to check whether a matrix entered by the user is a Sparse Matrix or not.

Ans)

Code:

```
#include <stdio.h>
int main()
{
int a[100][100],m,n;
printf("Enter the no of rows of a matrix: ");
scanf("%d",&m);
printf("Enter the no of columns of a matrix: ");
scanf("%d",&n);
printf("Enter the matrix elements: ");
for(int i=0;i<m;i++)
{
for(int j=0;j<n;j++)
{
scanf("%d",&a[i][j]);
}
}
int t=0;
for(int i=0;i<m;i++)
{
for(int j=0;j<n;j++)
{
if(a[i][j]==0)
t++;
}
}
printf("The given matrix is ");
t>=(m*n)/2?printf("a Sparse Matrix."):printf("not a Sparse Matrix");
return 0;
}
```

Output:

```
Enter the no of rows of a matrix: 5
Enter the no of columns of a matrix: 4
Enter the matrix elements: 1 0 2 0
                             3 0 0 0
                             0 0 9 2
                             0 4 0 0
                             0 3 0 5
The given matrix is a Sparse Matrix.

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

Q2) Write a program to merge two sorted arrays.

Ans)

Code:

```
#include <stdio.h>
void merge2arrays(int a[], int m,int b[],int n,int c[])
{
int i=0,j=0,k=0;
while(i<m && j<n)
{
if(a[i]<=b[j])
{
c[k]=a[i];
i++;
k++;
}
else
{
c[k]=b[j];
j++;
k++;
}
}
if(i==m)
{
while(j<n)
{
c[k]=b[j];
j++;
k++;
}
}
else
{
while(i<m)
{
c[k]=a[i];
i++;
k++;
}
}
}
int main()
{
int a[100],m,b[100],n,c[200];
printf("Enter the first array size: ");
scanf("%d",&m);
printf("Enter the first array elements (the array elements should be sorted in ascending order):
");
for(int i=0;i<m;i++)
{
```

```
scanf("%d",&a[i]);
}
printf("Enter the second array size: ");
scanf("%d",&n);
printf("Enter the second array elements (the array elements should be sorted in ascending order): ");
for(int i=0;i<n;i++)
{
scanf("%d",&b[i]);
}
merge2arrays(a,m,b,n,c);
printf("\nThe merged array is: ");
for(int i=0;i<m+n;i++)
{
printf("%d ",c[i]);
}
return 0;
}
```

Output:

```
Enter the first array size: 10
Enter the first array elements (the array elements should be sorted in ascending order): 1 12 2
3 34 45 56 67 78 89 90
Enter the second array size: 8
Enter the second array elements (the array elements should be sorted in ascending order): 19
20 34 48 59 68 82 104

The merged array is: 1 12 19 20 23 34 34 45 48 56 59 67 68 78 82 89 90 104

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

Q) WAP to perform insertion, deletion, traversal and updating an element in arrays.

Ans)

Code:

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

```
#include <stdlib.h>
int size=0;
void insert_at_beg(int a[],int ele)
{
if(size==99)
{
printf("\nThe array is full.");
return;
}
if(size==0)
{
a[0]=ele;
size++;
return;
}
for(int i=size-1;i>=0;i--)
{
a[i+1]=a[i];
}
a[0]=ele;
size++;
return;
}
void insert_at_any(int a[],int ele,int index)
{
if(size==99)
{
printf("\nThe array is full.");
return;
}
if(size==0)
{
a[0]=ele;
size++;
return;
}
for(int i=size-1;i>=index;i--)
{
a[i+1]=a[i];
}
a[index]=ele;
size++;
return;
}
void insert_at_end(int a[], int ele)
{
if(size==99)
{
printf("\nThe array is full.");
return;
}
if(size==0)
{
a[0]=ele;
size++;
}
a[size]=ele;
size++;
return;
}
```

```
void delete_at_beg(int a[])
{
if(size==0)
{
printf("\nThe array is empty.");
return;
}
if(size==1)
{
size--;
return;
}
for(int i=1;i<=size-1;i++)
{
a[i-1]=a[i];
}
size--;
return;
}
void delete_at_any(int a[],int index)
{
if(size==0)
{
printf("\nThe array is empty.");
return;
}
if(size==1)
{
size--;
return;
}
for(int i=index;i<size-1;i++)
{
a[i]=a[i+1];
}
size--;
return;
}
void delete_at_end(int a[])
{
if(size==0)
{
printf("\nThe array is empty.");
return;
}
size--;
return;
}
void traverse(int a[])
{
if(size==0)
{
printf("\nThe array is empty.");
return;
}
printf("The elements present in an array are: ");
for(int i=0;i<size;i++)
{
printf("%d ",a[i]);
}
```

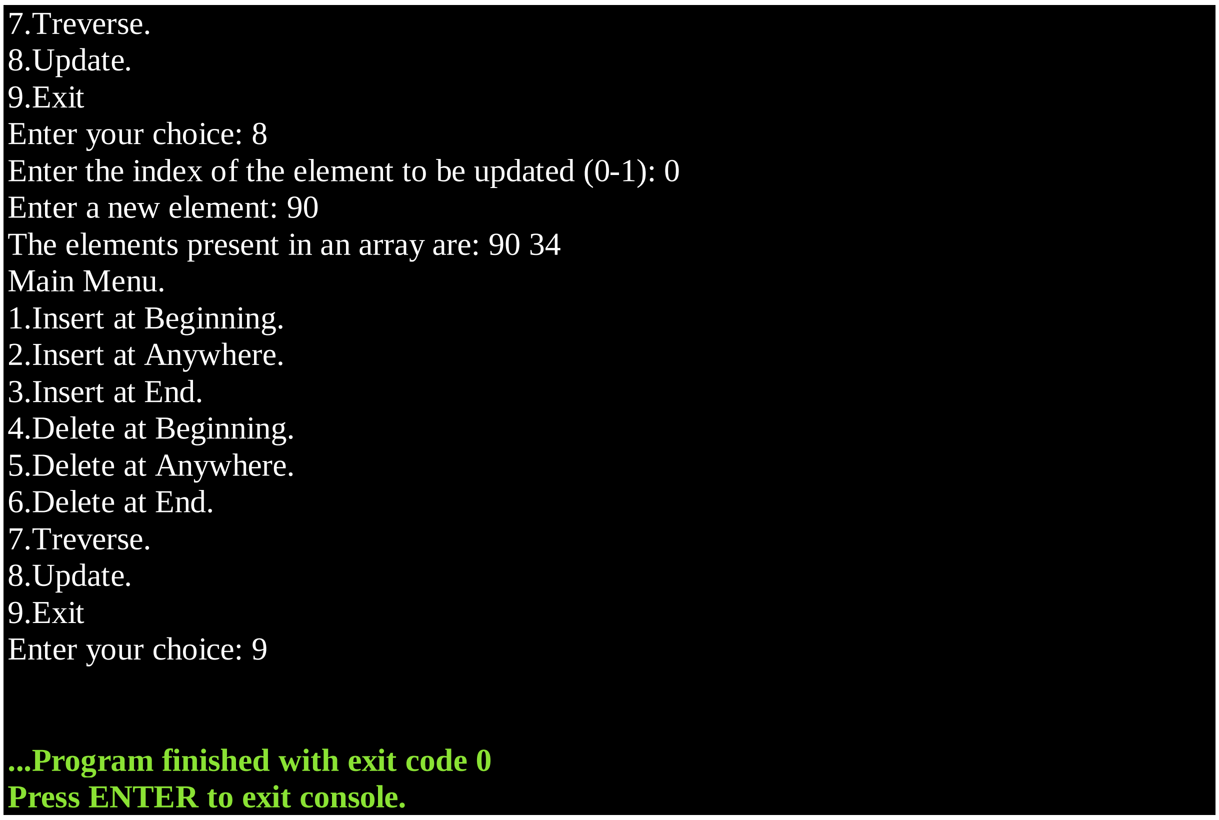
```
return;
}
void update(int a[],int ele,int index)
{
if(size==0)
{
printf("\nThe array is empty.");
return;
}
a[index]=ele;
return;
}
int main()
{
int choice;
int a[100];
while(1)
{
int index;
int ele;
printf("\nMain Menu.");
printf("\n1.Insert at Beginning.");
printf("\n2.Insert at Anywhere.");
printf("\n3.Insert at End.");
printf("\n4.Delete at Beginning.");
printf("\n5.Delete at Anywhere.");
printf("\n6.Delete at End.");
printf("\n7.Treverse.");
printf("\n8.Update.");
printf("\n9.Exit");
printf("\nEnter your choice: ");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("Enter an element to be inserted: ");
scanf("%d",&ele);
insert_at_beg(a,ele);
traverse(a);
break;
case 2:printf("Enter the index of the element to be inserted (1-%d): ",size-2);
scanf("%d",&index);
printf("Enter an element to be inserted: ");
scanf("%d",&ele);
insert_at_any(a,ele,index);
traverse(a);
break;
case 3:printf("Enter an element to be inserted: ");
scanf("%d",&ele);
insert_at_end(a,ele);
traverse(a);
break;
case 4:delete_at_beg(a);
traverse(a);
break;
case 5:printf("Enter the index of the element to be deleted (1-%d): ",size-2);
scanf("%d",&index);
delete_at_any(a,index);
traverse(a);
break;
case 6:delete_at_end(a);
traverse(a);
break;
case 7:traverse(a);
break;
case 8:printf("Enter the index of the element to be updated (0-%d): ",size-1);
scanf("%d",&index);
printf("Enter a new element: ");
scanf("%d",&ele);
update(a,ele,index);
traverse(a);
break;
case 9:exit(0);
}
}
return 0;
}
```

Output:

```
Main Menu.
1.Insert at Beginning.
2.Insert at Anywhere.
3.Insert at End.
4.Delete at Beginning.
5.Delete at Anywhere.
6.Delete at End.
7.Treverse.
8.Update.
9.Exit
Enter your choice: 1
Enter an element to be inserted: 34
The elements present in an array are: 34
Main Menu.
1.Insert at Beginning.
2.Insert at Anywhere.
3.Insert at End.
4.Delete at Beginning.
5.Delete at Anywhere.
6.Delete at End.
7.Treverse.
8.Update.
```

```
9.Exit
Enter your choice: 1
Enter an element to be inserted: 65
The elements present in an array are: 65 34
Main Menu.
1.Insert at Beginning.
2.Insert at Anywhere.
3.Insert at End.
4.Delete at Beginning.
5.Delete at Anywhere.
6.Delete at End.
7.Treverse.
8.Update.
9.Exit
Enter your choice: 1
Enter an element to be inserted: 67
The elements present in an array are: 67 65 34
Main Menu.
1.Insert at Beginning.
2.Insert at Anywhere.
3.Insert at End.
4.Delete at Beginning.
5.Delete at Anywhere.
6.Delete at End.
7.Treverse.
8.Update.
9.Exit
Enter your choice: 2
Enter the index of the element to be inserted (1-1): 1
Enter an element to be inserted: 45
The elements present in an array are: 67 45 65 34
Main Menu.
1.Insert at Beginning.
2.Insert at Anywhere.
3.Insert at End.
4.Delete at Beginning.
5.Delete at Anywhere.
6.Delete at End.
7.Treverse.
8.Update.
9.Exit
Enter your choice: 3
Enter an element to be inserted: 83
The elements present in an array are: 67 45 65 34 83
Main Menu.
1.Insert at Beginning.
2.Insert at Anywhere.
3.Insert at End.
4.Delete at Beginning.
5.Delete at Anywhere.
```

```
6.Delete at End.
7.Treverse.
8.Update.
9.Exit
Enter your choice: 4
The elements present in an array are: 45 65 34 83
Main Menu.
1.Insert at Beginning.
2.Insert at Anywhere.
3.Insert at End.
4.Delete at Beginning.
5.Delete at Anywhere.
6.Delete at End.
7.Treverse.
8.Update.
9.Exit
Enter your choice: 5
Enter the index of the element to be deleted (1-2): 1
The elements present in an array are: 45 34 83
Main Menu.
1.Insert at Beginning.
2.Insert at Anywhere.
3.Insert at End.
4.Delete at Beginning.
5.Delete at Anywhere.
6.Delete at End.
7.Treverse.
8.Update.
9.Exit
Enter your choice: 6
The elements present in an array are: 45 34
Main Menu.
1.Insert at Beginning.
2.Insert at Anywhere.
3.Insert at End.
4.Delete at Beginning.
5.Delete at Anywhere.
6.Delete at End.
7.Treverse.
8.Update.
9.Exit
Enter your choice: 7
The elements present in an array are: 45 34
Main Menu.
1.Insert at Beginning.
2.Insert at Anywhere.
3.Insert at End.
4.Delete at Beginning.
5.Delete at Anywhere.
6.Delete at End.
```


```
7.Treverse.
8.Update.
9.Exit
Enter your choice: 8
Enter the index of the element to be updated (0-1): 0
Enter a new element: 90
The elements present in an array are: 90 34
Main Menu.
1.Insert at Beginning.
2.Insert at Anywhere.
3.Insert at End.
4.Delete at Beginning.
5.Delete at Anywhere.
6.Delete at End.
7.Treverse.
8.Update.
9.Exit
Enter your choice: 9


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

Stacks and Queues

Q1) Write a program to perform following operations in stack:

(a) push

(b) pop

(c) peek

Ans)

Code:

```
#include <stdio.h>
int top=-1;
void peek(int stack[100]);
void push(int stack[100],int size,int val);
int pop(int stack[100]);
int main()
{
int size,val,del,choice;
int stack[100];
printf("\nEnter the maximum size of the stack: ");
scanf("%d",&size);
char ch='y';
do
{
printf("\nMain Menu.");
printf("\nChoose the option: ");
printf("\n1.Push");
printf("\n2.Pop");
printf("\n3.Peek");
scanf("%d",&choice);
switch(choice)
{
case 1: printf("\nEnter the value to be inserted: ");
scanf("%d",&val);
push(stack,size,val);
break;
case 2: del=pop(stack);
printf("\n%d is deleted from the stack.",del);
break;
case 3:peek(stack);
break;
default: printf("\nInvalid entry.");
}
printf("\nDo you want to continue?");
scanf("%s",&ch);
}while(ch=='y' || ch=='Y');
return 0;
}
void peek(int stack[])
{
if(top==-1)
{
printf("\nThe stack is underflowing.");
}
else
{
printf("\nThe elements in the top of the stack is: %d",stack[top]);
}
}
void push(int stack[],int size,int val)
{
if(top==size-1)
{
```

```
printf("\nThe stack is overflowing.");
}
else
{
top++;
stack[top]=val;
}
}
int pop(int stack[])
{
int temp;
if(top==-1)
{
printf("\nThe stack is underflowing.");
temp=-99999999;
}
else
{
temp=stack[top];
top--;
}
return temp;
}
```

Output:

```
Enter the maximum size of the stack: 5

Main Menu.
Choose the option:
1.Push
2.Pop
3.Peek
1

Enter the value to be inserted: 3
```

```
Do you want to continue?y

Main Menu.
Choose the option:
1.Push
2.Pop
3.Peek
1

Enter the value to be inserted: 6

Do you want to continue?y

Main Menu.
Choose the option:
1.Push
2.Pop
3.Peek
1

Enter the value to be inserted: 7

Do you want to continue?y

Main Menu.
Choose the option:
1.Push
2.Pop
3.Peek
3

The elements in the top of the stack is: 7
Do you want to continue?y

Main Menu.
Choose the option:
1.Push
2.Pop
3.Peek
2

7 is deleted from the stack.
Do you want to continue?y

Main Menu.
Choose the option:
1.Push
2.Pop
3.Peek
```

```
2

6 is deleted from the stack.
Do you want to continue?y

Main Menu.
Choose the option:
1.Push
2.Pop
3.Peek
2

3 is deleted from the stack.
Do you want to continue?n


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

Q2) Write a program to perform following operations in a stack:

(a) check for overflow condition

(b) check for underflow condition

(c) print all the elements

(d) print first and last element

Ans)

Code:

```
#include <stdio.h>

#include <stdlib.h>

int top=-1;

void peek(int stack[])

{

if(top==-1)

{

printf("\nThe stack is underflowing.");

}

else

{

printf("\nThe elements present in the top of the stack is: %d",stack[top]);
}
}
void push(int stack[],int size,int val)
{
if(top==size-1)
{
printf("\nThe stack is overflowing.");
}
else
{
top++;
stack[top]=val;
}
}
int pop(int stack[])
{
int temp;
if(top==-1)
{
printf("\nThe stack is underflowing.");
temp=-99999999;
}
else
{
temp=stack[top];
top--;
}
return temp;
}
void bottom(int stack[])
```

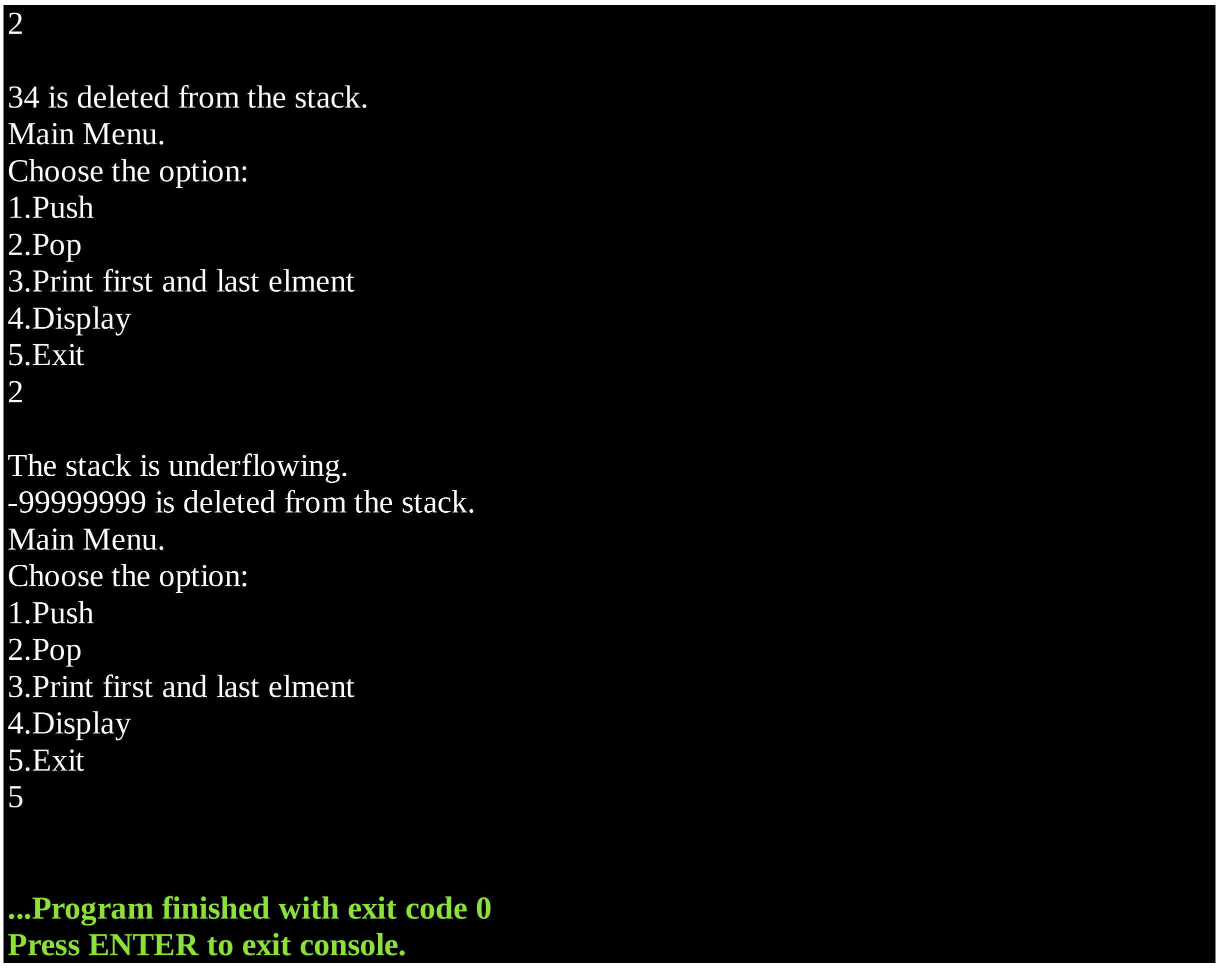
```
{
if(top==-1)
{
printf("\nThe stack is underflowing.");
}
else
{
printf("\nThe elements present in the bottom of the stack is: %d",stack[0]);
}
}
void display(int stack[])
{
if(top==-1)
{
printf("\nThe stack is underflowing.");
}
else
{
for(int i=0;i<=top;i++)
{
printf("%d ",stack[i]);
}
}
}
int main()
{
int size,val,del,choice;
int stack[100];
printf("\nEnter the maximum size of the stack: ");
scanf("%d",&size);
while(1)
{
printf("\nMain Menu.");
printf("\nChoose the option: ");
printf("\n1.Push");
printf("\n2.Pop");
printf("\n3.Print first and last elment");
printf("\n4.Display");
printf("\n5.Exit");
scanf("%d",&choice);
switch(choice)
{
case 1: printf("\nEnter the value to be inserted: ");
scanf("%d",&val);
push(stack,size,val);
break;
case 2: del=pop(stack);
printf("\n%d is deleted from the stack.",del);
break;
case 3: peek(stack);
bottom(stack);
break;
case 4: printf("\nThe elements present in the stack are: ");
display(stack);
break;
case 5: exit(0);
}
}
return 0;
}
```

Output:

```
Enter the maximum size of the stack: 5

Main Menu.
Choose the option:
1.Push
2.Pop
3.Print first and last elment
4.Display
5.Exit
1

Enter the value to be inserted: 34

Main Menu.
Choose the option:
1.Push
2.Pop
3.Print first and last elment
4.Display
5.Exit
1

Enter the value to be inserted: 45

Main Menu.
Choose the option:
1.Push
2.Pop
3.Print first and last elment
4.Display
5.Exit
1

Enter the value to be inserted: 56

Main Menu.
Choose the option:
1.Push
2.Pop
3.Print first and last elment
4.Display
5.Exit
3

The elements present in the top of the stack is: 56
The elements present in the bottom of the stack is: 34
Main Menu.
Choose the option:
```

```
1.Push
2.Pop
3.Print first and last elment
4.Display
5.Exit
3

The elements present in the top of the stack is: 56
The elements present in the bottom of the stack is: 34
Main Menu.
Choose the option:
1.Push
2.Pop
3.Print first and last elment
4.Display
5.Exit
4

The elements present in the stack are: 34 45 56
Main Menu.
Choose the option:
1.Push
2.Pop
3.Print first and last elment
4.Display
5.Exit
1

Enter the value to be inserted: 78

Main Menu.
Choose the option:
1.Push
2.Pop
3.Print first and last elment
4.Display
5.Exit
1

Enter the value to be inserted: 98

Main Menu.
Choose the option:
1.Push
2.Pop
3.Print first and last elment
4.Display
5.Exit
1
```

```
Enter the value to be inserted: 10

The stack is overflowing.
Main Menu.
Choose the option:
1.Push
2.Pop
3.Print first and last elment
4.Display
5.Exit
2

98 is deleted from the stack.
Main Menu.
Choose the option:
1.Push
2.Pop
3.Print first and last elment
4.Display
5.Exit
2

78 is deleted from the stack.
Main Menu.
Choose the option:
1.Push
2.Pop
3.Print first and last elment
4.Display
5.Exit
2

56 is deleted from the stack.
Main Menu.
Choose the option:
1.Push
2.Pop
3.Print first and last elment
4.Display
5.Exit
2

45 is deleted from the stack.
Main Menu.
Choose the option:
1.Push
2.Pop
3.Print first and last elment
4.Display
5.Exit
```



```
2

34 is deleted from the stack.
Main Menu.
Choose the option:
1.Push
2.Pop
3.Print first and last elment
4.Display
5.Exit
2

The stack is underflowing.
-99999999 is deleted from the stack.
Main Menu.
Choose the option:
1.Push
2.Pop
3.Print first and last elment
4.Display
5.Exit
5


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

Q3) Write a program to create a stack that can hold letters. Perform various operations like PUSH, POP, PEEK and TRAVERSE. Also check overflow and underflow condition.

Ans)

Code:

```
#include <stdio.h>

int top=-1;

int size;

void push(char stack[size],char value);

int pop(char stack[size]);

int peek(char stack[size]);

void traverse(char stack[size]);

int main()

{

printf("\nenter the maximum size of the stack");
```

```
scanf("%d",&size);
char stack[size];
char value;
int choice;
char c,d;
char ch='y';
do{
printf("\nMain Menu");
printf("\n1.Push");
printf("\n2.Pop");
printf("\n3.Peek");
printf("\n4.Traverse");
printf("\nEnter your choice");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the value to be entered");
scanf("%s",&value);
push(stack,value);
traverse(stack);
break;
case 2:d=pop(stack);
printf("\nthe element deleted from the stack is %c",d);
traverse(stack);
break;
case 3:c=peek(stack);
printf("\nthe current element entered in the stack is %c",c);
break;
case 4:traverse(stack);
break;
default:printf("\nInvalid Entry");
}
printf("\ndo you want to continue");
scanf("%s",&ch);
}while(ch=='y' || ch=='Y');
return 0;
}
void push(char stack[size],char value)
{
if(top==size-1)
{
printf("\nthe stack is overflowing");
}
else{
top=top+1;
stack[top]=value;
}
}
int pop(char stack[size])
{
if(top==-1)
{
printf("\nthe stack is underflow");
return '\0';
}
else
{
char temp;
temp=stack[top];
top--;
```

```
return temp;
}
}
int peek(char stack[size])
{
if(top==-1)
{
printf("\nthe stack is underflow");
return '\0';
}
else
{
return stack[top];
}
}
void traverse(char stack[size])
{
if(top==-1)
{
printf("\nthe stack is underflow");
}
else
{
printf("\nthe elements present in the stack are:");
printf("\n");
int i;
for(int i=top;i>=0;i--)
{
printf("%c ",stack[i]);
}
```

```
printf("\n");
}
}
```

Output:

```
enter the maximum size of the stack5

Main Menu
1.Push
2.Pop
3.Peek
4.Traverse
Enter your choice1
enter the value to be entereda

the elements present in the stack are:
a

do you want to continuey

Main Menu
1.Push
2.Pop
3.Peek
4.Traverse
Enter your choice1
enter the value to be enteredf

the elements present in the stack are:
f a

do you want to continuey

Main Menu
1.Push
2.Pop
3.Peek
4.Traverse
Enter your choice2

the element deleted from the stack is f
the elements present in the stack are:
a

do you want to continuey

Main Menu
1.Push
```

```
2.Pop
3.Peek
4.Traverse
Enter your choice1
enter the value to be enteredd

the elements present in the stack are:
d a

do you want to continuey

Main Menu
1.Push
2.Pop
3.Peek
4.Traverse
Enter your choice3

the current element entered in the stack is d
do you want to continuey

Main Menu
1.Push
2.Pop
3.Peek
4.Traverse
Enter your choice4

the elements present in the stack are:
d a

do you want to continuen


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

Q4) Write a program to print the reverse of the stack.

Ans)

Code:

```
#include <stdio.h>

int top=-1;

int size;

void push(char stack[size],char value);

int pop(char stack[size]);
int peek(char stack[size]);
void traverse(char stack[size]);
void reverse(char stack[size]);
int main()
{
printf("\nenter the maximum size of the stack");
scanf("%d",&size);
char stack[size];
char value;
int choice;
char c,d;
char ch='y';
do{
printf("\nMain Menu");
printf("\n1.Push");
printf("\n2.Pop");
printf("\n3.Peek");
printf("\n4.Traverse");
printf("\n5.Reverse");
printf("\nEnter your choice");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the value to be entered");
scanf("%s",&value);
push(stack,value);
traverse(stack);
break;
case 2:d=pop(stack);
printf("\nthe element deleted from the stack is %c",d);
traverse(stack);
break;
case 3:c=peek(stack);
printf("\nthe current element entered in the stack is %c",c);
break;
case 4:traverse(stack);
break;
case 5:reverse(stack);
traverse(stack);
break;
default:printf("\nInvalid Entry");
}
printf("\ndo you want to continue");
scanf("%s",&ch);
}while(ch=='y' || ch=='Y');
return 0;
}
void push(char stack[size],char value)
{
if(top==size-1)
{
printf("\nthe stack is overflowing");
}
else{
top=top+1;
stack[top]=value;
}
}
int pop(char stack[size])
{
```

```
if(top==-1)
{
printf("\nthe stack is underflow");
return '\0';
}
else
{
char temp;
temp=stack[top];
top--;
return temp;
}
}
int peek(char stack[size])
{
if(top==-1)
{
printf("\nthe stack is underflow");
return '\0';
}
else
{
return stack[top];
}
}
void traverse(char stack[size])
{
if(top==-1)
{
printf("\nthe stack is underflow");
```

```
}
else
{
printf("\nthe elements present in the stack are:");
printf("\n");
int i;
for(i=top;i>=0;i--)
{
printf("%c ",stack[i]);
}
printf("\n");
}
}
void reverse(char stack[size])
{
int i,j;
char temp;
if(top==-1)
{
printf("\n the reverse operation could not be performed");
}
else
{
for(i=0,j=top;i<j;i++,j--)
{
temp=stack[i];
stack[i]=stack[j];
stack[j]=temp;
}
//printf("%d",top);
```

```
}
}
```

Output:

```
enter the maximum size of the stack5

Main Menu
1.Push
2.Pop
3.Peek
4.Traverse
5.Reverse
Enter your choice1
enter the value to be entereda

the elements present in the stack are:
a

do you want to continuey

Main Menu
1.Push
2.Pop
3.Peek
4.Traverse
5.Reverse
Enter your choice1
enter the value to be enterede

the elements present in the stack are:
e a

do you want to continuey

Main Menu
1.Push
2.Pop
3.Peek
4.Traverse
5.Reverse
Enter your choice1
enter the value to be enteredy

the elements present in the stack are:
y e a

do you want to continuey

Main Menu
```

```
1.Push
2.Pop
3.Peek
4.Traverse
5.Reverse
Enter your choice5

the elements present in the stack are:
a e y

do you want to continuey

Main Menu
1.Push
2.Pop
3.Peek
4.Traverse
5.Reverse
Enter your choice3

the current element entered in the stack is a
do you want to continuey

Main Menu
1.Push
2.Pop
3.Peek
4.Traverse
5.Reverse
Enter your choice2

the element deleted from the stack is a
the elements present in the stack are:
e y

do you want to continuey

Main Menu
1.Push
2.Pop
3.Peek
4.Traverse
5.Reverse
Enter your choice5

the elements present in the stack are:
y e

do you want to continuey
```

```
Main Menu
1.Push
2.Pop
3.Peek
4.Traverse
5.Reverse
Enter your choice3

the current element entered in the stack is y
do you want to continuen


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

Q5) Write a program to solve Tower of Hanoi problem using recursion.

Ans)

Code:

```
#include <stdio.h>

int TowerofHanoi(int n);

void TOH(int n,char s,char a,char d);

int main()

{

int n;

printf("enter the no of disc to be moved");

scanf("%d",&n);

char s='a';

char a='b';

char d='c';

int moves=TowerofHanoi(n);

TOH(n,s,d,a);

printf("\n the no of moves is %d",moves);

return 0;

}

int TowerofHanoi(int n)

{
```

```
if(n==1)
{
return 1;
}
int c=2*TowerofHanoi(n-1)+1;
return c;
}
void TOH(int n,char s,char a,char d)
{
if(n==0)
{
return;
}
if(n==1)
{
printf("\n %c %c",s,d);
return;
}
TOH(n-1,s,d,a);
printf("\n %c %c",s,d);
TOH(n-1,a,s,d);
}
```

Output:

```
enter the no of disc to be moved3

a b
a c
b c
a b
c a
c b
a b
the no of moves is 7

...Program finished with exit code 0
```

```
Press ENTER to exit console.
```

Q6) Write a program to check nesting of parenthesis in an expression. Check the following conditions:

(a) The number of left parenthesis should be equal to the number of right parenthesis.

(b) Each right parenthesis is preceded by a matching left parenthesis.

Ans)

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#define size 50
int stack[size];
int top=-1;
void push(char p)
{
top++;
stack[top]=p;
}
void pop()
{
top--;
}
int peek()
{
return stack[top];
}
bool empty()
{
if(top==-1)
```

```
{
return true;
}
else
{
return false;
}
}
bool isBalanced(char expression[])
{
bool b=true;
if(expression[0]==')' || expression[0]==']' || expression[0]=='}')
{
b=false;
return b;
}
else
{
int len=strlen(expression);
for(int i=0;i<len;i++)
{
if(expression[i]=='(' || expression[i]=='[' || expression[i]=='{')
{
push(expression[i]);
}
else if(expression[i]==')')
{
char a=peek();
if(a=='(')
{
```

```
pop();
}
else
{
b=false;
return b;
}
}
else if(expression[i]==']')
{
char a=peek();
if(a=='[')
{
pop();
}
else
{
b=false;
return b;
}
}
else if(expression[i]=='}')
{
char a=peek();
if(a=='{')
{
pop();
}
else
{
```

```
false;
return b;
}
}
else
{
continue;
}
}
if(empty())
{
b=true;
return b;
}
else
{
b=false;
return b;
}
}
}
int main()
{
char input[40];
printf("Enter the expression (till 40 characters): ");
scanf("%s",&input);
bool b=isBalanced(input);
printf("\nThe expresion is ");
b?printf("balanced."):printf("not balanced");
return 0;
```

```
}
```

Output:

```
Enter the expression (till 40 characters): {a+[b*(c-d)]}/(f+g+h)

The expresion is balanced.

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

Q7) Write a program that accepts an infix expression and convert it into postfix expression. Also evaluate the postfix expression. You can assume that the entered expression is a valid expression is a valid infix expression.

Ans)

Code:

```
#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <string.h>

#include <math.h>

#include <ctype.h>

#define size 50

#define size2 50

char stack[size];

int top=-1;

void push(char p)

{

top++;

stack[top]=p;

}

void pop()

{

top--;

}
```

```
char peek()
{
return stack[top];
}
bool empty()
{
if(top==-1)
{
return true;
}
else
{
return false;
}
}
int stack2[size2];
int top2=-1;
void push2(int p)
{
top2++;
stack2[top2]=p;
}
void pop2()
{
top2--;
}
int peek2()
{
return stack2[top2];
}
```

```
bool empty2()
{
if(top2==-1)
{
return true;
}
else
{
return false;
}
}
void infix2postfix(char input[],char output[])
{
int len=strlen(input);
int k=0;
for(int i=0;i<len;i++)
{
if(input[i]=='(')
{
push(input[i]);
}
else if(input[i]==')')
{
while(!empty() && peek()!='(')
{
output[k]=peek();
pop();
k++;
}
pop();
```

```
}
else if(input[i]=='^')
{
char a=peek();
if(empty())
{
push(input[i]);
}
else if(a=='^')
{
output[k]=a;
pop();
k++;
while(!empty())
{
char b=peek();
if(b=='^')
{
output[k]=b;
pop();
k++;
}
else
{
break;
}

}
push(input[i]);
}
```

```
else
{
push(input[i]);
}
}
else if(input[i]=='*')
{
char a=peek();
if(empty())
{
push(input[i]);
}
else if(a=='^'||a=='/'||a=='*')
{
output[k]=a;
pop();
k++;
while(!empty())
{
char b=peek();
if(b=='^'||b=='/'||b=='*')
{
output[k]=b;
pop();
k++;
}
else
{
break;
}
```

```
}
push(input[i]);
}
else
{
push(input[i]);
}
}
else if(input[i]=='/')
{
char a=peek();
if(empty())
{
push(input[i]);
}
else if(a=='^'||a=='/'||a=='*')
{
output[k]=a;
pop();
k++;
while(!empty())
{
char b=peek();
if(b=='^'||b=='/'||b=='*')
{
output[k]=b;
pop();
k++;
}
```

```
else
{
break;
}


}
push(input[i]);
}
else
{
push(input[i]);
}
}
else if(input[i]=='+')
{
char a=peek();
if(empty())
{
push(input[i]);
}
else if(a=='^'||a=='/'||a=='*'||a=='+'||a=='-')
{
output[k]=a;
pop();
k++;
while(!empty())
{
char b=peek();
if(b=='^'||b=='/'||b=='*'||b=='+'||b=='-')
{
```

```
output[k]=b;
pop();
k++;
}
else
{
break;
}

}
push(input[i]);
}
else
{
push(input[i]);
}
}
else if(input[i]=='-')
{
char a=peek();
if(empty())
{
push(input[i]);
}
else if(a=='^'||a=='/'||a=='*'||a=='+'||a=='-')
{
output[k]=peek();
pop();
k++;
while(!empty())
```

```
{
char b=peek();
if(b=='^'||b=='/'||b=='*'||b=='+'||b=='-')
{
output[k]=b;
pop();
k++;
}
else
{
break;
}

}
push(input[i]);
}
else
{
push(input[i]);
}
}
else
{
output[k]=input[i];
k++;
}
}
while(!empty())
{
output[k]=peek();
```

```
pop();
k++;
}
}
int evapostfix(char output[])
{
int ans;
//int len=strlen(output);
//puts(output);
for(int i=0;output[i]!='\0';i++)
{
if(isdigit(output[i]))
{
push2(output[i]-48);
}
else
{
int a=peek2();
//printf("%d ",a);
pop2();
int b=peek2();
//printf("%d ",b);
pop2();
int c;
if(output[i]=='+')
{
c=b+a;
}
else if(output[i]=='-')
{
```

```
c=b-a;
}
else if(output[i]=='*')
{
c=b*a;
}
else if(output[i]=='/')
{
c=b/a;
}
else
{
c=pow(b,a);
}
push2(c);
}
}
ans=peek2();
pop2();
return ans;
}
int main()
{
char input[40],output[40];
printf("Enter the infix expression (till 40 characters): ");
scanf("%s",&input);
infix2postfix(input,output);
printf("\nThe postfix expression is: ");
for(int i=0;output[i]!='\0';i++)
{
```

```
printf("%c",output[i]);

}

int a=evapostfix(output);

printf("\nThe evaluated postfix expression is: %d",a);

return 0;

}
```

Output:

```
Enter the infix expression (till 40 characters): 1+2*(3^4)-5

The postfix expression is: 1234^*+5-
The evaluated postfix expression is: 158

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

Q8) Write a program to create a linear queue and perform various queue operations like ENQUEUE, DEQUEUE, PEEK and TRAVERSE. Also check for Overflow and Underflow condition.

Ans)

(a) Case 1:

Code:

```
#include <stdio.h>

int front=-1 ,rear=-1;

#define maxsize 10

void enqueue(int queue[maxsize],int value);

int dequeue(int queue[maxsize]);

int peek(int queue[maxsize]);

void traverse(int queue[maxsize]);

int main()

{

int queue[maxsize];

int value;

int choice;

int c,d;
char ch='y';
do{
printf("\nMain Menu");
printf("\n1.Enqueue");
printf("\n2.Dequeue");
printf("\n3.Peek");
printf("\n4.Traverse");
printf("\nEnter your choice");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the value to be entered");
scanf("%d",&value);
enqueue(queue,value);
break;
case 2:d=dequeue(queue);
printf("\nthe element deleted from the queue is %d",d);
break;
case 3:c=peek(queue);
printf("\nthe current element entered in the stack is %d",d);
break;
case 4:traverse(queue);
break;
default:printf("\nInvalid Entry");
}
printf("\ndo you want to continue");
scanf("%s",&ch);
}while(ch=='y' || ch=='Y');
return 0;
}
```

```
void enqueue(int queue[maxsize],int value)
{
if(rear==maxsize-1)
{
printf("\nthe queue is overflowing");
}
else{
if(front==-1)
{
front=0;
}
rear=rear+1;
queue[rear]=value;
}
}
int dequeue(int queue[maxsize])
{
if(front == rear+1)
{
printf("\nthe queue is underflow");
return '\0';
}
else
{
int temp;
temp=queue[front];
front++;
return temp;
}
}
```

```
int peek(int queue[maxsize])
{
if(front==rear+1)
{
printf("\nthe queue is underflow");
return '\0';
}
else
{
return queue[front];
}
}
void traverse(int queue[maxsize])
{
if(front==rear+1)
{
printf("\nthe queue is underflow");
}
else
{
printf("\nthe elements present in the queue are:");
printf("\n");
int i;
for(int i=front;i<=rear;i++)
{
printf("%d ",queue[i]);
}
printf("\n");
}
}
```

Output:

```
Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered5

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered3

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered
4
```

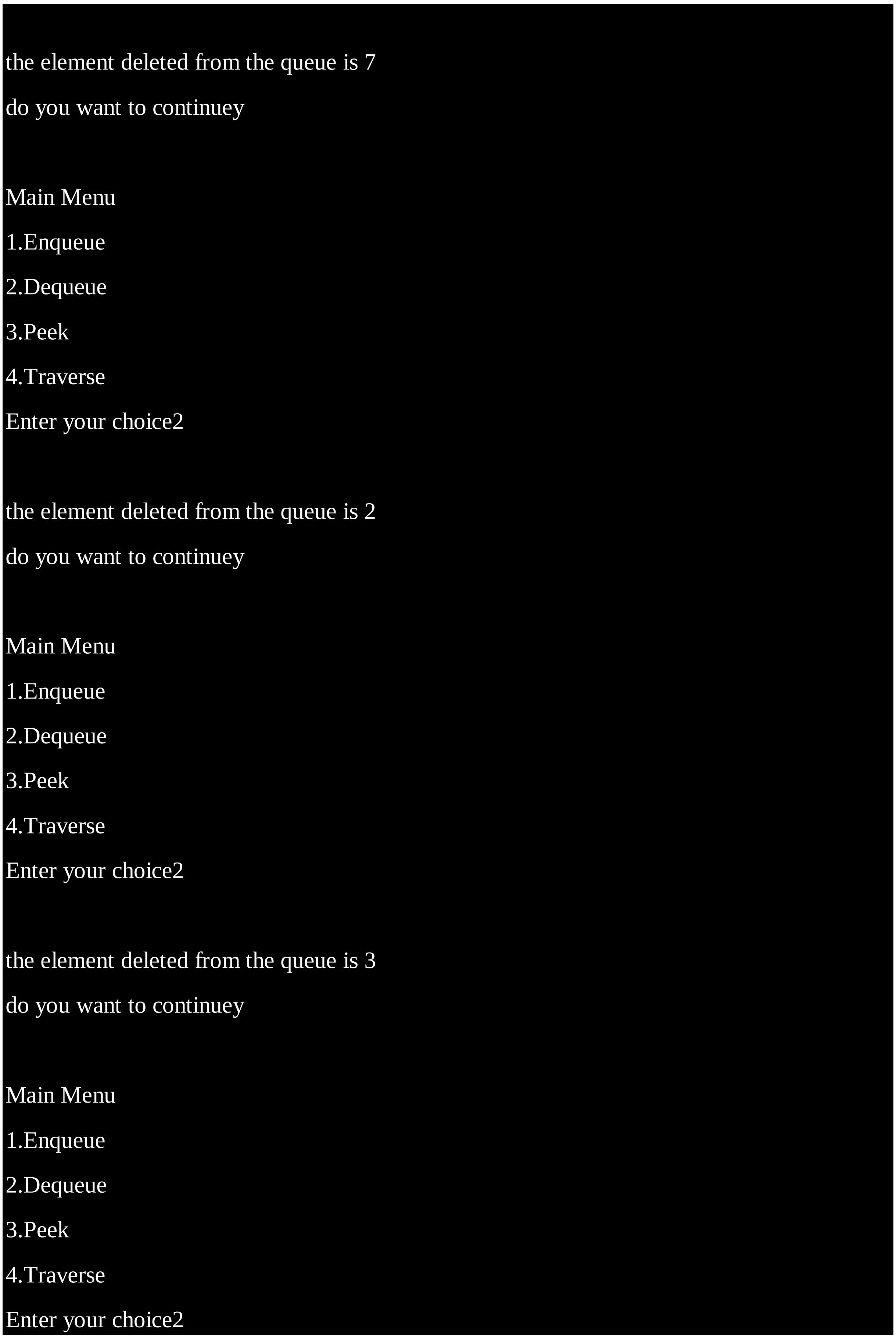
```
do you want to continuey



Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered3

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered6

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered7
```

```
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered2

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered3

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered3

do you want to continuey
```

```

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered8

do you want to continuey


Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered8

the queue is overflowing
do you want to continuey


Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice4

the elements present in the queue are:
```

```
5 3 4 3 6 7 2 3 3 8

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2

the element deleted from the queue is 5
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2

the element deleted from the queue is 3
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2
```

```

the element deleted from the queue is 4
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2

the element deleted from the queue is 3
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2

the element deleted from the queue is 6
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2
```


```

the element deleted from the queue is 7
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2

the element deleted from the queue is 2
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2

the element deleted from the queue is 3
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2
```

```

the element deleted from the queue is 3
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2

the element deleted from the queue is 8
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2

the queue is underflow
the element deleted from the queue is 0
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
```

```
Enter your choice4


the queue is underflow

do you want to continuey


Main Menu

1.Enqueue

2.Dequeue

3.Peek

4.Traverse

Enter your choice1

enter the value to be entered6


the queue is overflowing

do you want to continuen



...Program finished with exit code 0

Press ENTER to exit console.
```

(b) Case 2:

Code:

```
#include <stdio.h>

int front=-1 ,rear=-1;

#define maxsize 10

void enqueue(int queue[maxsize],int value);

int dequeue(int queue[maxsize]);

int peek(int queue[maxsize]);

void traverse(int queue[maxsize]);

int main()

{
```

```
int queue[maxsize];
int value;
int choice;
int c,d;
char ch='y';
do{
printf("\nMain Menu");
printf("\n1.Enqueue");
printf("\n2.Dequeue");
printf("\n3.Peek");
printf("\n4.Traverse");
printf("\nEnter your choice");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the value to be entered");
scanf("%d",&value);
enqueue(queue,value);
break;
case 2:d=dequeue(queue);
printf("\nthe element deleted from the queue is %d",d);
break;
case 3:c=peek(queue);
printf("\nthe current element entered in the stack is %d",d);
break;
case 4:traverse(queue);
break;
default:printf("\nInvalid Entry");
}
printf("\ndo you want to continue");
```

```
scanf("%s",&ch);
}while(ch=='y' || ch=='Y');
return 0;
}
void enqueue(int queue[maxsize],int value)
{
if(rear==maxsize-1)
{
printf("\nthe queue is overflowing");
}
else{
if(front==-1)
{
front=0;
}
rear=rear+1;
queue[rear]=value;
}
}
int dequeue(int queue[maxsize])
{
if(front==rear+1)
{
printf("\nthe queue is underflow");
return '\0';
}
else
{
int temp;
temp=queue[front];
```

```
int i;
for(i=0;i<rear;i++)
{
queue[i]=queue[i+1];
}
rear--;
return temp;
}
}
int peek(int queue[maxsize])
{
if(front==rear+1)
{
printf("\nthe queue is underflow");
return '\0';
}
else
{
return queue[front];
}
}
void traverse(int queue[maxsize])
{
if(front==rear+1)
{
printf("\nthe queue is underflow");
}
else
{
printf("\nthe elements present in the queue are:");
printf("\n");
int i;
for(int i=front;i<=rear;i++)
{
printf("%d ",queue[i]);
}
printf("\n");
}
}
```

Output :

```
Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered3

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered2

do you want to continuey
```

```
Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered4

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered5

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered6

do you want to continuey
```

```
Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered8

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered7

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered1

do you want to continuey
```

```
Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered9

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered1

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered2

the queue is overflowing
do you want to continuey
```

```
Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice4

the elements present in the queue are:
3 2 4 5 6 8 7 1 9 1

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2

the element deleted from the queue is 3
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2
```

```
the element deleted from the queue is 2
do you want to continuey


Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2


the element deleted from the queue is 4
do you want to continuey


Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2
the element deleted from the queue is 5
do you want to continuey


Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2


the element deleted from the queue is 6
```

```
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2

the element deleted from the queue is 8
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2

the element deleted from the queue is 7
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2

the element deleted from the queue is 1
```

```
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2

the element deleted from the queue is 9
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2

the element deleted from the queue is 1
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice2

the queue is underflow
the element deleted from the queue is 0
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice4

the queue is underflow
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice1
enter the value to be entered3

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
3.Peek
4.Traverse
Enter your choice4
```

```
the elements present in the queue are:
3


do you want to continuen



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

Q8) Write a program to implement ENQUEUE and DEQUEUE operations on a queue implementation in the form of the circular array.

Ans)

Code:

```
#include <stdio.h>
int front=-1 ,rear=-1;
#define maxsize 5
void enqueue(int cqueue[maxsize],int value);
int dequeue(int cqueue[maxsize]);
void traverse(int cqueue[maxsize]);
int main()
{
int cqueue[maxsize];
int value;
int choice;
int d;
char ch='y';
do{
printf("\nMain Menu");
printf("\n1.Enqueue");
printf("\n2.Dequeue");
//printf("\n3.Peek");
//printf("\n4.Traverse");
```

```
printf("\nEnter your choice");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the value to be entered");
scanf("%d",&value);
enqueue(cqueue,value);
traverse(cqueue);
break;
case 2:d=dequeue(cqueue);
printf("\nthe element deleted from the circular queue is %d",d);
traverse(cqueue);
break;
default:printf("\nInvalid Entry");
}
printf("\ndo you want to continue");
scanf("%s",&ch);
}while(ch=='y' || ch=='Y');
return 0;
}
void enqueue(int cqueue[maxsize],int value)
{
if((rear==maxsize-1 && front==0)||(front==rear+1))
{
printf("\nthe circular queue is overflowing");
}
else
{
if(front==-1)
{
```

```
front=0;
rear=0;
}
else if(rear==maxsize-1)
{
rear=0;
}
else
{
rear=rear+1;
}
cqueue[rear]=value;
}
}
int dequeue(int cqueue[maxsize])
{
if(front==-1 && rear==-1)
{
printf("\nthe circular queue is underflow");
return 0;
}
else
{
int temp;
if(front == rear)
{
temp=cqueue[front];
front =rear=-1;
}
else if(front==maxsize-1)
```

```
{
temp=cqueue[front];
front=0;
}
else
{
temp=cqueue[front];
front++;
}
return temp;
}
}
void traverse(int cqueue[maxsize])
{
if(front==-1 && rear==-1)
{
printf("\nthe circular queue is underflow");
}
else
{
printf("\nthe elements present in the circular queue are:");
printf("\n");
int i;
if(front<=rear)
{
for(int i=front;i<=rear;i++)
{
printf("%d ",cqueue[i]);
}
printf("\n");
}
else
{
for(int i=front;i<=maxsize-1;i++)
{
printf("%d ",cqueue[i]);
}
for(int i=0;i<=rear;i++)
{
printf("%d ",cqueue[i]);
}
printf("\n");
}
}
}
```

Output:

```
Main Menu
1.Enqueue
2.Dequeue
Enter your choice1
enter the value to be entered45

the elements present in the circular queue are:
45

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
```

```
Enter your choice1
enter the value to be entered34

the elements present in the circular queue are:
45 34

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
Enter your choice1
enter the value to be entered94

the elements present in the circular queue are:
45 34 94

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
Enter your choice63

Invalid Entry
do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
```

```
Enter your choice1
enter the value to be entered63

the elements present in the circular queue are:
45 34 94 63

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
Enter your choice1
enter the value to be entered72

the elements present in the circular queue are:
45 34 94 63 72

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
Enter your choice1
enter the value to be entered28

the circular queue is overflowing
the elements present in the circular queue are:
45 34 94 63 72

do you want to continuey
```

```

Main Menu
1.Enqueue
2.Dequeue
Enter your choice2

the element deleted from the circular queue is 45
the elements present in the circular queue are:
34 94 63 72

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
Enter your choice2

the element deleted from the circular queue is 34
the elements present in the circular queue are:
94 63 72

do you want to continuey

Main Menu
1.Enqueue
2.Dequeue
Enter your choice1
enter the value to be entered45

the elements present in the circular queue are:
```

```
94 63 72 45


do you want to continuen



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

Q9) Write a program to identify whether a string is a palindrome using dequeue.

Ans)

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
char queue [50];
int front=0;
int rear=-1;
int t=0;
void push(char p)
{
rear=(rear+1)%50;
queue[rear]=p;
t++;
}
void pop1()
{
front=(front+1)%50;
t--;
}
void pop2()
{
```

```
rear=(rear-1)%50;
t--;
}
char peek1()
{
return queue[front];
}
char peek2()
{
return queue[rear];
}
bool check()
{
return front>=rear? true : false;
}
bool palindrome(char a[])
{
int n=0;
for(int i=0;a[i]!='\0';i++)
{
n++;
}
for(int i=0;a[i]!='\0';i++)
{
push(a[i]);
}
while(!check())
{
char b=peek1();
char c=peek2();
if(b==c)
{
pop1();
pop2();
}
else
{
break;
}
}
if(n%2==0 && t==0)
{
return true;
}
else if(n%2!=0 && t==1)
{
return true;
}
else
{
return false;
}
}
int main()
{
char a[40];
printf("Enter the string: ");
scanf("%s",&a);
bool b=palindrome(a);
if(b)
```

```
{
printf("\nIt is a palindrome.");
}
else
{
printf("\nIt is not a palindrome.");
}
return 0;
}
```

Output:

```
Enter the string: kanak

It is a palindrome.

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

Linked List

Q1) Write a program to simply create a linked list and perform following operations:

(a) Insert at the beginning

(b) Insert at the end

(c) Insert at a specific node

(d) Delete the first node

(e) Delete the last node

(f) Delete the specific node

(g) find the length of the list.

Ans)

Code:

```
//Just dont mind, I have combined all the cases asked in the question in one. Sorry.
//Basic Linked List implementation
#include <stdio.h>
#include <stdlib.h>
struct Node
{
int data;
struct Node *next;//stores the addrerss of the node
};
struct Node *head=NULL;
struct Node *tail=NULL;
void takeinput();//inseting the node in the linked list
void insert();//inserting ith node in the linked list
void delete();//deleting ith node in the linked list
void length();//finding the length of the linked list
void print();//printing the linked list
int main()
{
int data,pos,delpos;
int choice;
while(1)
{
printf("\n Main Menu");
printf("\n 1.Create");
printf("\n 2.Insert");
printf("\n 3.Delete");
printf("\n 4.Display");
printf("\n 5.Length");
printf("\n 6.Exit");
printf("\n Enter your choice:");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the data in the linked list before it implementing it");
takeinput();
print();
break;
case 2:insert();
print();
break;
case 3:delete();
print();
break;
case 4:print();
break;
case 5:length();
break;
case 6:exit(0);
break;
default: printf("invalid entry");
}
}
return 0;
}
void takeinput()
{
struct Node *n=(struct Node*) malloc(sizeof(struct Node));
int data;
printf("\n enter the data");
scanf("%d",&n->data);
n->next=NULL;
if(head==NULL)
{
head=n;
tail=n;
}
else
{
//tail=head;
tail->next=n;
tail=tail->next;
}
}
void insert()
{
int pos;
struct Node *n=(struct Node*) malloc(sizeof(struct Node));
printf("enter the position in which the data is to be inserted");
scanf("%d",&pos);
```

```
printf("enter the data");
scanf("%d",&n->data);
n->next=NULL;
struct Node *temp;
int ctr=0;
if(pos==0)
{
n->next=head;
head=n;
}
temp=head;
while(temp!=NULL && ctr<pos-1)
{
temp=temp->next;
ctr++;
}
if(temp!=NULL)
{
struct Node *a=temp->next;
temp->next=n;
n->next=a;
}
}
void delete()
{
	int delpos;
	if(head==NULL)
	{
		printf("pls enter the element in the linked list");
	return;
```

```
        }
printf("enter the position in which the data is to be deleted");
scanf("%d",&delpos);
int ctr=0;
struct Node *temp=head;
if(delpos==0)
{
head=temp->next;
temp->next=NULL;
free(temp);
return;
}
while(temp->next!=NULL && ctr<delpos-1)
{
temp=temp->next;
ctr++;
}
if(temp->next!=NULL)
{
struct Node *a=temp->next;
struct Node *b=a->next;
temp->next=b;
a->next=NULL;
free(a);
}
return;
}
void print()
{
struct Node *temp=head;
```

```
if(temp==NULL)
{
printf("\n pls enter any enement in the linked list");
return;
}
while(temp!=NULL)
{
printf("%d ",temp->data);
temp=temp->next;
}
printf("\n");
return;
}
void length()
{
struct Node *temp=head;
int i=0;
while(temp!=NULL)
{
i++;
temp=temp->next;
}
printf("\n the length of the linked list is %d",i);
}
```

Output:

```
Main Menu
1.Create
2.Insert
3.Delete
4.Display
```

```
5.Length

6.Exit

Enter your choice:1

enter the data in the linked list before it implementing it

enter the data1

1


Main Menu

1.Create

2.Insert

3.Delete

4.Display

5.Length

6.Exit

Enter your choice:2

enter the position in which the data is to be inserted1

enter the data2

1 2


Main Menu

1.Create

2.Insert

3.Delete

4.Display

5.Length

6.Exit

Enter your choice:2

enter the position in which the data is to be inserted0

enter the data3

3 1 2
```

```
Main Menu
1.Create
2.Insert
3.Delete
4.Display
5.Length
6.Exit
Enter your choice:2
enter the position in which the data is to be inserted2
enter the data4
3 1 4 2

Main Menu
1.Create
2.Insert
3.Delete
4.Display
5.Length
6.Exit
Enter your choice:3
enter the position in which the data is to be deleted1
3 4 2

Main Menu
1.Create
2.Insert
3.Delete
4.Display
5.Length
```

```
6.Exit
Enter your choice:3
enter the position in which the data is to be deleted0
4 2

Main Menu
1.Create
2.Insert
3.Delete
4.Display
5.Length
6.Exit
Enter your choice:4
4 2

Main Menu
1.Create
2.Insert
3.Delete
4.Display
5.Length
6.Exit
Enter your choice:5

the length of the linked list is 2
Main Menu
1.Create
2.Insert
3.Delete
4.Display
```

```
5.Length
6.Exit
Enter your choice:6



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

Q2) Write a program to perform the following operations in the linked list:

(a) Delete the node of the given data.

(b) Delete the next node of the given data.

Ans)

Code:

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
int data;
struct Node *next;//stores the addrerss of the node
};
struct Node *head=NULL;
struct Node *tail=NULL;
void takeinput();//inseting the node in the linked list
void insert();//inserting ith node in the linked list
void print();//printing the linked list
int search(int ele);//searching the element in the linked list
void delete1();//deleting node data of the linked list
void delete2();//deleting next node of the linked list
int main()
{
```

```
int data,pos,delpos;
int choice;
while(1)
{
printf("\n Main Menu");
printf("\n 1.Create");
printf("\n 2.Insert");
printf("\n 3.Display");
printf("\n 4.Delete node data");
printf("\n 5.Delete next node");
printf("\n 6.Exit");
printf("\n Enter your choice:");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the data in the linked list before it implementing it");
takeinput();
print();
break;
case 2:insert();
print();
break;
case 3:print();
break;
case 4:delete1();
print();
break;
case 5:delete2();
print();
break;
case 6:exit(0);
break;
default: printf("invalid entry");
}
}
return 0;
}
void takeinput()
{
struct Node *n=(struct Node*) malloc(sizeof(struct Node));
int data;
printf("\n enter the data");
scanf("%d",&n->data);
n->next=NULL;
if(head==NULL)
{
head=n;
tail=n;
}
else
{
//tail=head;
tail->next=n;
tail=tail->next;
}
}
void insert()
{
int pos;
struct Node *n=(struct Node*) malloc(sizeof(struct Node));
```

```
printf("enter the position in which the data is to be inserted");
scanf("%d",&pos);
printf("enter the data");
scanf("%d",&n->data);
n->next=NULL;
struct Node *temp;
int ctr=0;
if(pos==0)
{
n->next=head;
head=n;
}
temp=head;
while(temp!=NULL && ctr<pos-1)
{
temp=temp->next;
ctr++;
}
if(temp!=NULL)
{
struct Node *a=temp->next;
temp->next=n;
n->next=a;
}
}
/*void delete()
{
int delpos;
if(head==NULL)
{
```

```
printf("pls enter the element in the linked list");
return;
}
printf("enter the position in which the data is to be deleted");
scanf("%d",&delpos);
int ctr=0;
struct Node *temp=head;
if(delpos==0)
{
head=temp->next;
temp->next=NULL;
free(temp);
return;
}
while(temp->next!=NULL && ctr<delpos-1)
{
temp=temp->next;
ctr++;
}
if(temp->next!=NULL)
{
struct Node *a=temp->next;
struct Node *b=a->next;
temp->next=b;
a->next=NULL;
free(a);
}
return;
}*/
void print()
```

```
{
struct Node *temp=head;
if(temp==NULL)
{
printf("\n pls enter any enement in the linked list");
return;
}
while(temp!=NULL)
{
printf("%d ",temp->data);
temp=temp->next;
}
printf("\n");
return;
}
int search(int ele)
{
int ctr=0;
struct Node *temp=head;
while(temp!=NULL)
{
if(temp->data==ele)
{
return ctr;
}
ctr++;
temp=temp->next;
}
return -1;
}
```

```
void delete1()
{
int ele;
if(head==NULL)
{
printf("pls enter the element in the linked list");
return;
}
printf("enter the element to be deleted");
scanf("%d",&ele);
int ctr=0;
struct Node *temp=head;
int x=search(ele);
if(x==-1)
{
printf("No such element is found");
return;
}
else if(x==0)
{
head=temp->next;
temp->next=NULL;
free(temp);
return;
}
else
{
int ctr=0;
while(temp->next!=NULL && ctr<x-1)
{
```

```
temp=temp->next;
ctr++;
}
if(temp->next!=NULL)
{
struct Node *a=temp->next;
struct Node *b=a->next;
temp->next=b;
a->next=NULL;
free(a);
}
}
return;
}
void delete2()
{
int ele;
if(head==NULL)
{
printf("pls enter the element in the linked list");
return;
}
printf("enter the element for which next element is to be deleted");
scanf("%d",&ele);
int ctr=0;
struct Node *temp=head;
int x=search(ele)+1;
if(x==-1)
{
printf("No such element is found");
```

```
return;
}
else if(x==0)
{
head=temp->next;
temp->next=NULL;
free(temp);
return;
}
else
{
int ctr=0;
while(temp->next!=NULL && ctr<x-1)
{
temp=temp->next;
ctr++;
}
if(temp->next!=NULL)
{
struct Node *a=temp->next;
struct Node *b=a->next;
temp->next=b;
a->next=NULL;
free(a);
}
}
return;
}
```

Output:

```
Main Menu
```

```
1.Create
2.Insert
3.Display
4.Delete node data
5.Delete next node
6.Exit
Enter your choice:1
enter the data in the linked list before it implementing it
enter the data4
4

Main Menu
1.Create
2.Insert
3.Display
4.Delete node data
5.Delete next node
6.Exit
Enter your choice:2
enter the position in which the data is to be inserted3
enter the data4
4

Main Menu
1.Create
2.Insert
3.Display
4.Delete node data
5.Delete next node
6.Exit
```

```
Enter your choice:2
enter the position in which the data is to be inserted0
enter the data3
3 4

Main Menu
1.Create
2.Insert
3.Display
4.Delete node data
5.Delete next node
6.Exit
Enter your choice:2
enter the position in which the data is to be inserted1
enter the data9
3 9 4

Main Menu
1.Create
2.Insert
3.Display
4.Delete node data
5.Delete next node
6.Exit
Enter your choice:2
enter the position in which the data is to be inserted3
enter the data3
3 9 4 3

Main Menu
```

```
1.Create
2.Insert
3.Display
4.Delete node data
5.Delete next node
6.Exit
Enter your choice:4
enter the element to be deleted9
3 4 3

Main Menu
1.Create
2.Insert
3.Display
4.Delete node data
5.Delete next node
6.Exit
Enter your choice:5
enter the element for which next element is to be deleted4
3 4

Main Menu
1.Create
2.Insert
3.Display
4.Delete node data
5.Delete next node
6.Exit
Enter your choice:6
```

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

Q3) Write a program to append the first node to the last in the linked list.

Ans)

Code:

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
int data;
struct Node *next;//stores the addrerss of the node
};
struct Node *head=NULL;
struct Node *tail=NULL;
void takeinput();//inseting the node in the linked list
void insert();//inserting ith node in the linked list
void print();//printing the linked list
struct Node *append();//appending the first node to the last
int length();
int main()
{
int data,pos,delpos;
int choice;
while(1)
{
printf("\n Main Menu");
printf("\n 1.Create");
printf("\n 2.Insert");
printf("\n 3.Display");
printf("\n 4.Append first to last");
//printf("\n 5.Delete next node");
printf("\n 5.Exit");
printf("\n Enter your choice:");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the data in the linked list before it implementing it");
takeinput();
print();
break;
case 2:insert();
print();
break;
case 3:print();
break;
case 4:head=append();
print();
break;
case 5:exit(0);
break;
default: printf("invalid entry");
}
}
return 0;
}
void takeinput()
{
struct Node *n=(struct Node*) malloc(sizeof(struct Node));
```

```
int data;
printf("\n enter the data");
scanf("%d",&n->data);
n->next=NULL;
if(head==NULL)
{
head=n;
tail=n;
}
else
{
//tail=head;
tail->next=n;
tail=tail->next;
}
}
void insert()
{
int pos;
struct Node *n=(struct Node*) malloc(sizeof(struct Node));
printf("enter the position in which the data is to be inserted");
scanf("%d",&pos);
printf("enter the data");
scanf("%d",&n->data);
n->next=NULL;
struct Node *temp;
int ctr=0;
if(pos==0)
{
n->next=head;
head=n;
}
temp=head;
while(temp!=NULL && ctr<pos-1)
{
temp=temp->next;
ctr++;
}
if(temp!=NULL)
{
struct Node *a=temp->next;
temp->next=n;
n->next=a;
}
}
int length()
{
//Write your code here
int i=0;
struct Node *temp=head;
while(temp!= NULL)
{
i++;
temp=temp->next;
}
return i;
}

/*int length()
{
```

```
if(head==NULL)
{
return 0;
}
struct Node *temp=head;
int a=length(temp->next);
return a+1;
}*/

/*void delete()
{
int delpos;
if(head==NULL)
{
printf("pls enter the element in the linked list");
return;
}
printf("enter the position in which the data is to be deleted");
scanf("%d",&delpos);
int ctr=0;
struct Node *temp=head;
if(delpos==0)
{
head=temp->next;
temp->next=NULL;
free(temp);
return;
}
while(temp->next!=NULL && ctr<delpos-1)
{
```

```
temp=temp->next;
ctr++;
}
if(temp->next!=NULL)
{
struct Node *a=temp->next;
struct Node *b=a->next;
temp->next=b;
a->next=NULL;
free(a);
}
return;
}*/
void print()
{
struct Node *temp=head;
if(temp==NULL)
{
printf("\n pls enter any enement in the linked list");
return;
}
while(temp!=NULL)
{
printf("%d ",temp->data);
temp=temp->next;
}
printf("\n");
return;
}
/*int search(int ele)
```

```
{
int ctr=0;
struct Node *temp=head;
while(temp!=NULL)
{
if(temp->data==ele)
{
return ctr;
}
ctr++;
temp=temp->next;
}
return -1;
}
void delete1()
{
int ele;
if(head==NULL)
{
printf("pls enter the element in the linked list");
return;
}
printf("enter the element to be deleted");
scanf("%d",&ele);
int ctr=0;
struct Node *temp=head;
int x=search(ele);
if(x==-1)
{
printf("No such element is found");
return;
}
else if(x==0)
{
head=temp->next;
temp->next=NULL;
free(temp);
return;
}
else
{
int ctr=0;
while(temp->next!=NULL && ctr<x-1)
{
temp=temp->next;
ctr++;
}
if(temp->next!=NULL)
{
struct Node *a=temp->next;
struct Node *b=a->next;
temp->next=b;
a->next=NULL;
free(a);
}
}
return;
}
void delete2()
{
```

```
int ele;
if(head==NULL)
{
printf("pls enter the element in the linked list");
return;
}
printf("enter the element for which next element is to be deleted");
scanf("%d",&ele);
int ctr=0;
struct Node *temp=head;
int x=search(ele)+1;
if(x==-1)
{
printf("No such element is found");
return;
}
else if(x==0)
{
head=temp->next;
temp->next=NULL;
free(temp);
return;
}
else
{
int ctr=0;
while(temp->next!=NULL && ctr<x-1)
{
temp=temp->next;
ctr++;
}
if(temp->next!=NULL)
{
struct Node *a=temp->next;
struct Node *b=a->next;
temp->next=b;
a->next=NULL;
free(a);
}
}
return;
}*/
struct Node *append()
{
//struct Node *h=head;
int n;
printf("enter the no of nodes to be appended");
scanf("%d",&n);
struct Node *temp=head;
int len=length();
int ctr=len-n;
if(n!=0 && n<len && len!=0)
{
int i=1;
while(i<ctr)
{
temp=temp->next;
i++;
}
struct Node *h2=temp->next;
```

```
temp->next=NULL;
int j=1;
struct Node *t2=h2;
while(j<n)
{
t2=t2->next;
j++;
}
t2->next=head;
return h2;
}
else
{
return head;
}
}
```

Output:

```
Main Menu
1.Create
2.Insert
3.Display
4.Append first to last
5.Exit
Enter your choice:1
enter the data in the linked list before it implementing it
enter the data6
6


Main Menu
1.Create
```

```
2.Insert
3.Display
4.Append first to last
5.Exit
Enter your choice:1
enter the data in the linked list before it implementing it
enter the data7
6 7

Main Menu
1.Create
2.Insert
3.Display
4.Append first to last
5.Exit
Enter your choice:1
enter the data in the linked list before it implementing it
enter the data4
6 7 4

Main Menu
1.Create
2.Insert
3.Display
4.Append first to last
5.Exit
Enter your choice:1
enter the data in the linked list before it implementing it
enter the data8
6 7 4 8
```

```
Main Menu
1.Create
2.Insert
3.Display
4.Append first to last
5.Exit
Enter your choice:4
enter the no of nodes to be appended2
4 8 6 7

Main Menu
1.Create
2.Insert
3.Display
4.Append first to last
5.Exit
Enter your choice:5



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

Q4) Write a program to implement the count method, which should return a count of the number of times the given item is found in the list.

Ans)

Code:

```
#include <stdio.h>
#include <stdlib.h>
struct node
```

```
{
int data;
struct node *next;
};
struct node *head=NULL;
struct node *tail=NULL;
void insert();
void display();
int count(int ele);
int main()
{
int ele;
int n;
printf("\n Enter the no of elements that you wan to insert in the linked list: ");
scanf("%d",&n);
int i;
for(i=0;i<n;i++)
{
insert();
}
printf("\n The elements present in the linked list are: ");
display();
printf("\n Enter the element whoe occurance has to be counted in the linked list: ");
scanf("%d",&ele);
int occurance=count(ele);
printf("\n %d has occured %d times in the linked list.",ele,occurance);
return 0;
}
void insert()
{
```

```
//struct node *tail=head;
int data;
printf("\n Enter the element in the linked list");
scanf("%d",&data);
struct node *newnode=(struct node*)malloc(sizeof(struct node));
newnode->data=data;
newnode->next=NULL;
if(head==NULL)
{
head=tail=newnode;
}
else
{
tail->next=newnode;
tail=tail->next;
}
}
void display()
{
struct node *temp=head;
printf("\n");
while(temp!=NULL)
{
printf("%d ",temp->data);
temp=temp->next;
}
}
int count(int ele)
{
int ctr=0;
```

```
struct node *temp=head;
while(temp!=NULL)
{
if(temp->data==ele)
{
ctr++;
}
temp=temp->next;
}
return ctr;
}
```

Output:

```
Enter the no of elements that you wan to insert in the linked list: 9

Enter the element in the linked list2

Enter the element in the linked list12

Enter the element in the linked list12

Enter the element in the linked list34

Enter the element in the linked list56

Enter the element in the linked list18

Enter the element in the linked list23

Enter the element in the linked list45

Enter the element in the linked list1

The elements present in the linked list are:
2 12 12 34 56 18 23 45 1
Enter the element whoe occurance has to be counted in the linked list: 12

12 has occured 2 times in the linked list.

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

Q5) Write a program to reverse the elements of a linked list.

Ans)

Code:

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
int data;
struct node *next;
};
struct node *head=NULL;
struct node *tail=NULL;
void insert();
void display();
struct node *reverse();
int main()
{
//int ele;
int n;
printf("\n Enter the no of elements that you wan to insert in the linked list: ");
scanf("%d",&n);
int i;
for(i=0;i<n;i++)
{
insert();
}
printf("\n The elements present in the linked list are: ");
display();
printf("\n The reversed linked list is: ");
head=reverse();
```

```
display();
return 0;
}
void insert()
{
//struct node *tail=head;
int data;
printf("\n Enter the element in the linked list");
scanf("%d",&data);
struct node *newnode=(struct node*)malloc(sizeof(struct node));
newnode->data=data;
newnode->next=NULL;
if(head==NULL)
{
head=tail=newnode;
}
else
{
tail->next=newnode;
tail=tail->next;
}
}
void display()
{
struct node *temp=head;
printf("\n");
while(temp!=NULL)
{
printf("%d ",temp->data);
temp=temp->next;
}
}
struct node *reverse()
{
struct node *curr=head;
struct node *prev=NULL;
struct node *n=curr->next;
while(curr!=NULL)
{
curr->next=prev;
prev=curr;
curr=n;
if(curr)
n=curr->next;
}
return prev;
}
```

Output:

```
Enter the no of elements that you wan to insert in the linked list: 9

Enter the element in the linked list23

Enter the element in the linked list45

Enter the element in the linked list12

Enter the element in the linked list34

Enter the element in the linked list56

Enter the element in the linked list76

Enter the element in the linked list90

Enter the element in the linked list12

Enter the element in the linked list34
```

```
The elements present in the linked list are:
23 45 12 34 56 76 90 12 34
The reversed linked list is:
34 12 90 76 56 34 12 45 23

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

Q6) Write a program to implement stacks using linked list.

Ans)

Code:

```
//Stack Implementation Using Linked List
#include <stdio.h>
#include <stdlib.h>
struct stack
{
int data;
struct stack *next;
};
struct stack *top=NULL;
//struct node *tail=NULL;
void push();
void pop();
void peek();
void display();
//struct node *reverse();
int main()
{
int choice;
while(1)
{
printf("\n Main Menu");
printf("\n 1.Push");
printf("\n 2.Pop");
printf("\n 3.Peek");
printf("\n 4.Display");
printf("\n 5.Exit");
printf("\n Enter your choice: ");
scanf("%d",&choice);
switch(choice)
{
case 1:push();
display();
break;
case 2:pop();
display();
break;
case 3:peek();
break;
case 4:display();
break;
case 5:exit(0);
break;
default: printf("Invalid Entry");
}
}
return 0;
}
void push()
{
//struct node *tail=head;
int data;
printf("\n Enter the element in the linked list.");
```

```
scanf("%d",&data);

struct stack *newele=(struct stack*)malloc(sizeof(struct stack));

newele->data=data;

newele->next=top;

top=newele;

}

void pop()

{

if(top==NULL)

{

printf("\n Stack is overflowing.");

return;

}

struct stack *temp=top;

top=temp->next;

temp->next=NULL;

free(temp);


}

void peek()

{

if(top==NULL)

{

printf("\n Stack is overflowing.");

return;

}

printf("\n The element present on the top of the stack is: %d",top->data);

}

void display()

{
```

```
if(top==NULL)
{
printf("\n Stack is overflowing.");
return;
}
struct stack *temp=top;
printf("\n");
while(temp!=NULL)
{
printf("%d ",temp->data);
temp=temp->next;
}
}
```

Output:

```
Main Menu
1.Push
2.Pop
3.Peek
4.Display
5.Exit
Enter your choice: 1

Enter the element in the linked list.23

23
Main Menu
1.Push
2.Pop
3.Peek
4.Display
5.Exit
Enter your choice: 1

Enter the element in the linked list.23

23 23
Main Menu
1.Push
2.Pop
3.Peek
4.Display
```

```
5.Exit
Enter your choice: 1

Enter the element in the linked list.345

345 23 23
Main Menu
1.Push
2.Pop
3.Peek
4.Display
5.Exit
Enter your choice: 1

Enter the element in the linked list.56

56 345 23 23
Main Menu
1.Push
2.Pop
3.Peek
4.Display
5.Exit
Enter your choice: 1

Enter the element in the linked list.94

94 56 345 23 23
Main Menu
1.Push
2.Pop
3.Peek
4.Display
5.Exit
Enter your choice: 1

Enter the element in the linked list.48

48 94 56 345 23 23
Main Menu
1.Push
2.Pop
3.Peek
4.Display
5.Exit
Enter your choice: 1

Enter the element in the linked list.721

721 48 94 56 345 23 23
```

```
Main Menu
1.Push
2.Pop
3.Peek
4.Display
5.Exit
Enter your choice: 2

48 94 56 345 23 23
Main Menu
1.Push
2.Pop
3.Peek
4.Display
5.Exit
Enter your choice: 2

94 56 345 23 23
Main Menu
1.Push
2.Pop
3.Peek
4.Display
5.Exit
Enter your choice: 3

The element present on the top of the stack is: 94
Main Menu
1.Push
2.Pop
3.Peek
4.Display
5.Exit
Enter your choice: 4

94 56 345 23 23
Main Menu
1.Push
2.Pop
3.Peek
4.Display
5.Exit
Enter your choice: 5


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

Q7) Write a program to implement queues using linked list.

Ans)

Code:

```
//Queue Implementation Using Linked List
#include <stdio.h>
#include <stdlib.h>
struct queue
{
int data;
struct queue *next;
};
struct queue *front=NULL;
struct queue *rear=NULL;
void Enqueue();
void Dequeue();
//void peek();
void display();
//struct node *reverse();
int main()
{
int choice;
while(1)
{
printf("\n Main Menu");
printf("\n 1.Enqueue");
printf("\n 2.Dequeue");
printf("\n 3.Display");
printf("\n 4.Exit");
printf("\n Enter your choice: ");
scanf("%d",&choice);
```

```
switch(choice)
{
case 1:Enqueue();
display();
break;
case 2:Dequeue();
display();
break;
case 3:display();
break;
case 4:exit(0);
break;
default: printf("Invalid Entry");
}
}
return 0;
}
void Enqueue()
{
//struct node *tail=head;
int data;
printf("\n Enter the element in the linked list.");
scanf("%d",&data);
struct queue *newele=(struct queue*)malloc(sizeof(struct queue));
newele->data=data;
newele->next=NULL;
if(front==NULL)
{
front=newele;
rear=newele;
```

```
}
else
{
//tail=head;
rear->next=newele;
rear=rear->next;
}
}
void Dequeue()
{
if(front==NULL)
{
printf("\n Queue is underflowing.");
return;
}
struct queue *temp=front;
front=temp->next;
temp->next=NULL;
free(temp);
}
void display()
{
if(front==NULL)
{
printf("\n Queue is underflowing.");
return;
}
struct queue *temp=front;
printf("\n");
while(temp!=NULL)
```

```
{
printf("%d ",temp->data);
temp=temp->next;
}
}
```

Output:

```
Main Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 1

Enter the element in the linked list.2

2
Main Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 1

Enter the element in the linked list.34

2 34
Main Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 1

Enter the element in the linked list.498

2 34 498
Main Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 1

Enter the element in the linked list.347

2 34 498 347
```

```
Main Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 1

Enter the element in the linked list.267

2 34 498 347 267
Main Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 1

Enter the element in the linked list.529

2 34 498 347 267 529
Main Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit


Enter your choice: 1

Enter the element in the linked list.54

2 34 498 347 267 529 54
Main Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 1

Enter the element in the linked list.3

2 34 498 347 267 529 54 3
Main Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 2
```

```
34 498 347 267 529 54 3
Main Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 2

498 347 267 529 54 3
Main Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 3

498 347 267 529 54 3
Main Menu
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 4


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

Q8) Write a program to perform addition of polynomial expressions using linked list.

Ans)

Code:

```
#include <stdio.h>

#include <stdlib.h>

struct polynomial

{

int coefficient;

int degree;

struct polynomial *next;

};

struct polynomial *pt=NULL;
struct polynomial *insert(struct polynomial *p);
void display(struct polynomial *p);
struct polynomial *polysum(struct polynomial *p1,struct polynomial *p2);
int main()
{
int m,n;
struct polynomial *p1=NULL;
struct polynomial *p2=NULL;
struct polynomial *p=NULL;
printf("\n Enter the no of terms of the first polynomial: ");
scanf("%d",&m);
printf("\n Enter the no of terms of the second polynomial: ");
scanf("%d",&n);
printf("\n Enter the first polynomial:");
for(int i=0;i<m;i++)
{
struct polynomial *pa=insert(p1);
p1=pa;
}
pt=NULL;
printf("\n Enter the second polynomial:");
for(int i=0;i<n;i++)
{
struct polynomial *pb=insert(p2);
p2=pb;
}
printf("\n The first polynomial is: p1(x)= ");
display(p1);
printf("\n The second polynomial is: p2(x)= ");
display(p2);
```

```
printf("\n The polynomial sum is: p(x)= ");
p=polysum(p1,p2);
display(p);
return 0;
}
struct polynomial *insert(struct polynomial *p)
{
int coefficient;
int degree;
printf("\n Enter the coefficient: ");
scanf("%d",&coefficient);
printf("\n Enter the degree: ");
scanf("%d",&degree);
struct polynomial *newpoly=(struct polynomial*)malloc(sizeof(struct polynomial));
newpoly->coefficient=coefficient;
newpoly->degree=degree;
newpoly->next=NULL;
if(p==NULL)
{
p=pt=newpoly;
}
else
{
pt->next=newpoly;
pt=newpoly;
}
return p;
}
void display(struct polynomial *p)
{
```

```
struct polynomial *temp=p;
//printf("\n");
while(temp->next!=NULL)
{
printf("%dx%d + ",temp->coefficient,temp->degree);
temp=temp->next;
}
printf("%dx%d",temp->coefficient,temp->degree);
}
struct polynomial *polysum(struct  polynomial *p1,struct polynomial *p2)
{
struct polynomial *p=NULL;
struct polynomial *t=NULL;
struct polynomial *t1=p1;
struct polynomial *t2=p2;
while(t1!=NULL && t2!=NULL)
{
if(t1->degree<t2->degree)
{
if(p==NULL)
{
p=t=t2;
}
else
{
t->next=t2;
t=t2;
}
t2=t2->next;
}
```

```
else if(t1->degree>t2->degree)
{
if(p==NULL)
{
p=t=t1;
}
else
{
t->next=t1;
t=t1;
}
t1=t1->next;
}
else
{
struct polynomial *n=(struct polynomial*)malloc(sizeof(struct polynomial));
n->degree=t1->degree;
n->coefficient=t1->coefficient+t2->coefficient;
n->next=NULL;
if(p==NULL)
{
p=t=n;
}
else
{
t->next=n;
t=n;
}
t1=t1->next;
t2=t2->next;
}
}
if(t1==NULL)
{
while(t2!=NULL)
{
t->next=t2;
t=t2;
t2=t2->next;
}
}
else
{
while(t1!=NULL)
{
t->next=t1;
t=t1;
t1=t1->next;
}
}
return p;
}
```

Output:

```
Enter the no of terms of the first polynomial: 5

Enter the no of terms of the second polynomial: 3

Enter the first polynomial:
Enter the coefficient: 4

Enter the degree: 5

Enter the coefficient: 5

Enter the degree: 4
Enter the coefficient: 2

Enter the degree: 3

Enter the coefficient: 3

Enter the degree: 2

Enter the coefficient: 7

Enter the degree: 1

Enter the second polynomial:
Enter the coefficient: 9

Enter the degree: 6

Enter the coefficient: 6

Enter the degree: 4

Enter the coefficient: 3

Enter the degree: 2

The first polynomial is: p1(x)= 4x5 + 5x4 + 2x3 + 3x2 + 7x1
The second polynomial is: p2(x)= 9x6 + 6x4 + 3x2
The polynomial sum is: p(x)= 9x6 + 4x5 + 11x4 + 2x3 + 6x2 + 7x1

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

Q9) WAP to perform insertion deletion and treversal of doubly linked list.

Ans)

Code:

```
#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node *next;

struct node *prev;
```

```
};
struct node *head=NULL;
struct node *tail=NULL;
int length();
void Takeinput();
void Insert();
void Delete();
void Display();
int main()
{
int choice;
while(1)
{
printf("\nMain Menu");
printf("\n1.Creation of the List");
printf("\n2.Insert");
printf("\n3.Delete");
printf("\n4.Display");
printf("\n5.Exit");
printf("\nEnter your choice");
scanf("%d",&choice);
switch (choice)
{
case 1:Takeinput();
Display();
break;
case 2:Insert();
Display();
break;
case 3:Delete();
Display();
break;
case 4:Display();
break;
case 5:exit(0);
break;
default:printf("\n Invalid Entry");
}
}
}
int length()
{
int l=0;
struct node *temp=head;
while(temp!=NULL)
{
l++;
temp=temp->next;
}
return l;
}
void Takeinput()
{
int data;
struct node *n=(struct node *)malloc(sizeof(struct node));
printf("\nEnter the data");
scanf("%d",&data);
n->data=data;
n->next=NULL;
n->prev=NULL;
```

```
if(head==NULL)
{
head=tail=n;
}
else
{
tail->next=n;
n->prev=tail;
tail=n;
}
}
void Insert()
{
int l=length();
int data;
struct node *n=(struct node *)malloc(sizeof(struct node));
int pos;
printf("\nEnter the position at which the node is to be inserted");
scanf("%d",&pos);
if(pos>l)
{
printf("\nInvalid Location");
return;
}
printf("\nEnter the data");
scanf("%d",&data);
n->data=data;
n->next=NULL;
n->prev=NULL;
struct node *temp=head;
```

```
if(pos==0)

{

n->next=head;

head->prev=n;

head=n;

}

else if(pos==l)

{

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=n;

n->prev=temp;

}

else

{

int i=0;

while(i<pos-1 && temp!=NULL)

{

temp=temp->next;

i++;

}

if(temp!=NULL)

{

n->next=temp->next;

temp->next->prev=n;

n->prev=temp;

temp->next=n;

}
```

```
}
}
void Delete()
{
int l=length();
struct node *temp=head;
int pos;
if(head==NULL)
{
return;
}
else if(head->next==NULL)
{
head=NULL;
free(head);
}
else
{
printf("\nEnter the position at which the node is to be deleted");
scanf("%d",&pos);
if(pos>=l)
{
printf("\nInvalid position");
}
else if(pos==0)
{
temp=head;
head=head->next;
head->prev=NULL;
free(temp);
}
else if(pos==l-1)
{
while(temp->next!=NULL)
{
temp=temp->next;
}
temp->prev->next=NULL;
free(temp);
}
else
{
int i=0;
while(i<pos && temp->next!=NULL)
{
temp=temp->next;
i++;
}
if(temp->next!=NULL)
{
temp->prev->next=temp->next;
temp->next->prev=temp->prev;
free(temp);
}
}
}
}
void Display()
{
struct node *temp=head;
```

```
if(head==NULL)
{
printf("\nThe list is empty");
}
printf("\n");
while(temp!=NULL)
{
printf("%d ",temp->data);
temp=temp->next;
}
}
```

Output:

```
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice1

Enter the data34

34
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice1
Enter the data475

34 475
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice2

Enter the position at which the node is to be inserted0

Enter the data23

23 34 475
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice2

Enter the position at which the node is to be inserted9

Invalid Location
23 34 475
Main Menu
```

```
1.Creation of the List
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice2

Enter the position at which the node is to be inserted2

Enter the data190

23 34 190 475
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice2

Enter the position at which the node is to be inserted4

Enter the data64

23 34 190 475 64
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
```

```
5.Exit
Enter your choice3

Enter the position at which the node is to be deleted0

34 190 475 64
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice2

Enter the position at which the node is to be inserted0

Enter the data78

78 34 190 475 64
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice3

Enter the position at which the node is to be deleted6

Invalid position
```

```
78 34 190 475 64
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice3

Enter the position at which the node is to be deleted2

78 34 475 64
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice2

Enter the position at which the node is to be inserted3

Enter the data56

78 34 475 56 64
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
```

```
5.Exit
Enter your choice2

Enter the position at which the node is to be inserted0


Enter the data1

1 78 34 475 56 64
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice3

Enter the position at which the node is to be deleted5

1 78 34 475 56
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice4

1 78 34 475 56
Main Menu
```

```
1.Creation of the List

2.Insert

3.Delete

4.Display

5.Exit

Enter your choice5



...Program finished with exit code 0

Press ENTER to exit console.
```

Q10) WAP to reverse a doubly linked list.

Ans)

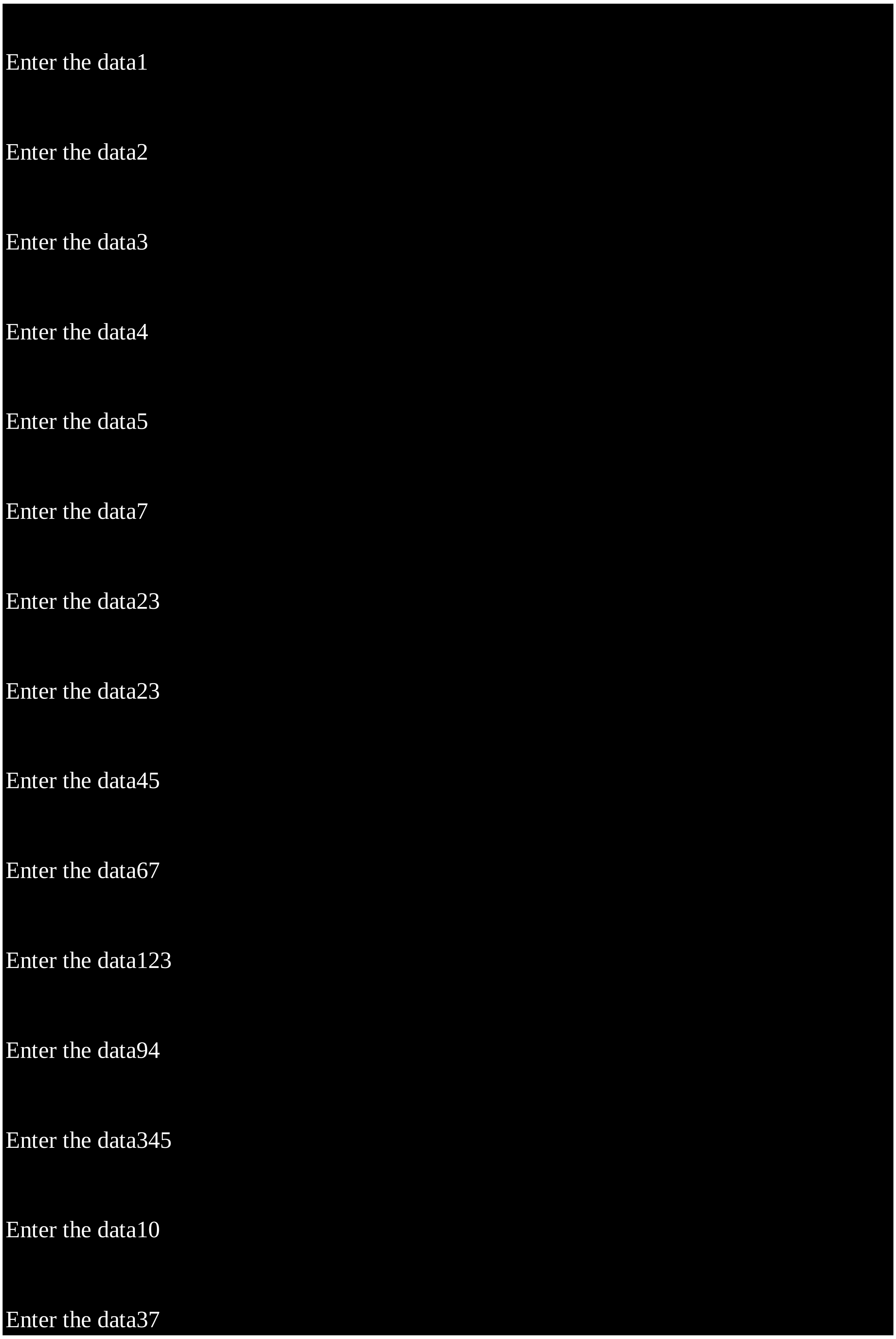
Code:

```
#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node *next;

struct node *prev;

};

struct node *head=NULL;

struct node *tail=NULL;

void insert();

void display();

struct node *reverse();

int main()

{

int n;

printf("\n Enter the no of elements that you wan to insert in the linked list: ");
```

```
scanf("%d",&n);
int i;
for(i=0;i<n;i++)
{
insert();
}
printf("\n The elements present in the linked list are: ");
display();
printf("\n The reversed linked list is: ");
head=reverse();
display();
return 0;
}
void insert()
{
int data;
struct node *n=(struct node *)malloc(sizeof(struct node));
printf("\nEnter the data");
scanf("%d",&data);
n->data=data;
n->next=NULL;
n->prev=NULL;
if(head==NULL)
{
head=tail=n;
}
else
{
tail->next=n;
n->prev=tail;
tail=n;
}
}
void display()
{
struct node *temp=head;
printf("\n");
while(temp!=NULL)
{
printf("%d ",temp->data);
temp=temp->next;
}
}
struct node *reverse()
{
struct node *curr=head;
struct node *p=NULL;
struct node *n=curr->next;
while(curr!=NULL)
{
curr->next=p;
p=curr;
curr=n;
if(curr)
n=curr->next;
}
return p;
}
```

Output:

```
Enter the no of elements that you wan to insert in the linked list: 20
```



```
Enter the data1

Enter the data2

Enter the data3

Enter the data4

Enter the data5

Enter the data7

Enter the data23

Enter the data23

Enter the data45

Enter the data67

Enter the data123

Enter the data94

Enter the data345

Enter the data10

Enter the data37
Enter the data2


Enter the data34


Enter the data56


Enter the data23


Enter the data45


The elements present in the linked list are:
1 2 3 4 5 7 23 23 45 67 123 94 345 10 37 2 34 56 23 45
The reversed linked list is:
45 23 56 34 2 37 10 345 94 123 67 45 23 23 7 5 4 3 2 1


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

Q11) WAP to create, display, insert and delete the doubly linked list.

Ans)

Code:

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
int data;
struct Node *next;//stores the addrerss of the node
};
struct Node *head=NULL;
struct Node *tail=NULL;
void takeinput();//inserting the node in the linked list
void insert();//inserting ith node in the linked list
void delete();//deleting ith node in the linked list
int length();//finding the length of the linked list
void print();//printing the linked list
int main()
{
int data,pos,delpos;
int choice;
while(1)
{
printf("\n Main Menu");
printf("\n 1.Create");
printf("\n 2.Insert");
printf("\n 3.Delete");
printf("\n 4.Display");
printf("\n 5.Exit");
printf("\n Enter your choice:");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the data in the linked list before it implementing it");
takeinput();
print();
break;
case 2:insert();
print();
break;
case 3:delete();
print();
break;
case 4:print();
break;
case 5:exit(0);
break;
default: printf("invalid entry");
}
}
return 0;
}
void takeinput()
{
struct Node *n=(struct Node*) malloc(sizeof(struct Node));
int data;
printf("\n enter the data");
scanf("%d",&n->data);
n->next=NULL;
if(head==NULL)
{
n->next=n;
head=tail=n;
}
else
{
//tail=head;
tail->next=n;
n->next=head;
tail=n;
}
```

```
}
int length()
{
struct Node *temp=head;
int i=0;
do
{
temp=temp->next;
i++;
}while(temp!=head);
return i;
}
void insert()
{
int l=length();
int pos;
printf("enter the position in which the data is to be inserted");
scanf("%d",&pos);
if(pos>l)
{
printf("\nInvalid position.");
return;
}
struct Node *n=(struct Node*) malloc(sizeof(struct Node));
printf("enter the data");
scanf("%d",&n->data);
n->next=NULL;
struct Node *temp;
int ctr=0;
if(pos==0)
```

```
{
n->next=head;
head=n;
tail->next=head;
}
else if(pos==l)
{
tail->next=n;
tail=n;
tail->next=head;
}
else
{
temp=head;
while(ctr<pos-1)
{
temp=temp->next;
ctr++;
}
if(temp!=NULL)
{
struct Node *a=temp->next;
temp->next=n;
n->next=a;
}
}
}
void delete()
{
int delpos;
```

```
if(head==NULL)
{
printf("pls enter the element in the linked list");
return;
}
int l=length();
printf("enter the position in which the data is to be deleted");
scanf("%d",&delpos);
if(delpos>=l)
{
printf("\nInvalid position.");
return;
}
int ctr=0;
struct Node *temp=head;
if(delpos==0)
{
head=temp->next;
temp->next=NULL;
tail->next=head;
free(temp);
return;
}
else if(delpos==l-1)
{
while(ctr<delpos-1)
{
temp=temp->next;
ctr++;
}
```

```
struct Node *a=tail;
a->next=NULL;
tail=temp;
tail->next=head;
free(a);
}
else
{
while(ctr<delpos-1)
{
temp=temp->next;
ctr++;
}
if(temp->next!=NULL)
{
struct Node *a=temp->next;
struct Node *b=a->next;
temp->next=b;
a->next=NULL;
free(a);
}
}
return;
}
void print()
{
struct Node *temp=head;
if(temp==NULL)
{
printf("\n pls enter any enement in the linked list");
```

```
return;
}
printf("\n");
do
{
printf("%d ",temp->data);
temp=temp->next;
}while(temp!=head);
printf("%d ",temp->data);
return;
}
```

Output:

```
Main Menu
1.Create
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice:1
enter the data in the linked list before it implementing it
enter the data1

1 1
Main Menu
1.Create
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice:2
enter the position in which the data is to be inserted0
enter the data2

2 1 2
Main Menu
1.Create
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice:2
enter the position in which the data is to be inserted1
enter the data3

2 3 1 2
Main Menu
1.Create
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice:2
enter the position in which the data is to be inserted3
enter the data4

2 3 1 4 2
Main Menu
1.Create
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice:4

2 3 1 4 2
Main Menu
1.Create
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice:3
enter the position in which the data is to be deleted0

3 1 4 3
Main Menu
1.Create
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice:3
enter the position in which the data is to be deleted2

3 1 3
Main Menu
1.Create
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice:1
```

```
enter the data in the linked list before it implementing it
enter the data5


3 1 5 3
Main Menu
1.Create
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice:3
enter the position in which the data is to be deleted1

3 5 3
Main Menu
1.Create
2.Insert
3.Delete
4.Display
5.Exit
Enter your choice:5


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

Q12) WAP to perform insertion and deletion operations in double circular linked list.

Ans)

Code:

```
#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node *next;

struct node *prev;

};

struct node *head=NULL;

struct node *tail=NULL;

void takeinput()
```

```
{
struct node *newnode=(struct node*)malloc(sizeof(struct node));
printf("\nEnter the data");
scanf("%d",&newnode->data);
newnode->prev=NULL;
newnode->next=NULL;
if(head==NULL)
{
head=tail=newnode;
newnode->prev=head;
newnode->next=head;
}
else
{
tail->next=newnode;
newnode->prev=tail;
head->prev=newnode;
newnode->next=head;
tail=newnode;
}
}
int length()
{
int l=0;
struct  node *temp=head;
do{
l++;
temp=temp->next;
}while(temp!=head);
return l;
```

```
}
void insert()
{
int l=length();
int pos;
printf("\nEnter the position at which you want to insert the node (0th position to the %dth position of the linked list)",l);
scanf("%d",&pos);
if(pos>l)
{
printf("\nInvalid loaction");
return;
}
struct node *newnode=(struct node*)malloc(sizeof(struct node));
printf("\nEnter the data");
scanf("%d",&newnode->data);
newnode->prev=NULL;
newnode->next=NULL;
if(pos==0)
{
head->prev->next=newnode;
newnode->prev=head->prev;
head->prev=newnode;
newnode->next=head;
head=newnode;
}
/*else if(pos==l)
{
tail->next=newnode;
newnode->prev=tail;
head->prev=newnode;
newnode->next=head;
tail=newnode;
}*/
else
{
int ctr=0;
struct node *temp=head;
while(ctr<pos-1)
{
temp=temp->next;
ctr++;
}
temp->next->prev=newnode;
newnode->next=temp->next;
temp->next=newnode;
newnode->prev=temp;
}
}
void delete()
{
if(head==NULL)
{
printf("\n Pls enter atleast one element");
return;
}
int l=length();
int pos;
printf("\nEnter the position at which you want to delete the node (0th position to the %dth position of the linked list)",l-1);
scanf("%d",&pos);
if(pos>=l)
```

```
{
printf("\nInvalid loaction");
return;
}
struct node *temp=head;
if(pos==0)
{
temp=head;
head->prev->next=head->next;
head->next->prev=head->prev;
head=head->next;
temp->next=NULL;
temp->prev=NULL;
free(temp);
}
/*else if(pos==l-1)
{
tail->next=newnode;
newnode->prev=tail;
head->prev=newnode;
newnode->next=head;
tail=newnode;
}*/
else
{
int ctr=0;
while(ctr<pos)
{
temp=temp->next;
ctr++;
}
temp->prev->next=temp->next;
temp->next->prev=temp->prev;
temp->next=NULL;
temp->prev=NULL;
free(temp);
}
}
void display()
{
if(head==NULL)
{
printf("\n Pls enter atleast one element");
return;
}
struct  node *temp=head;
printf("\n");
do{
printf("%d ",temp->data);
temp=temp->next;
}while(temp!=head);
printf("%d",temp->data);
}
int main()
{
int choice;
while(1)
{
printf("\nMain Menu");
printf("\n1.Creation of the List");
printf("\n2.Insert");
printf("\n3.Delete");
printf("\n4.Display");
printf("\n5.Exit");
printf("\nNote: You can use this option only before deletion of the last node of the circular doubly linked list.");
printf("\nEnter your choice");
scanf("%d",&choice);
switch (choice)
{
case 1:takeinput();
display();
break;
case 2:insert();
display();
break;
case 3:delete();
display();
break;
case 4:display();
break;
case 5:exit(0);
break;
default:printf("\n Invalid Entry");
}
}
}
```

Output:

```
Main Menu
1.Creation of the List
2.Insert
3.Delete
```

```
4.Display
5.Exit
Note: You can use this option only before deletion of the last node of the circular doubly link
ed list.
Enter your choice1

Enter the data1

1 1
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
5.Exit
Note: You can use this option only before deletion of the last node of the circular doubly link
ed list.
Enter your choice1

Enter the data2

1 2 1
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
5.Exit
Note: You can use this option only before deletion of the last node of the circular doubly link
ed list.
Enter your choice1

Enter the data3

1 2 3 1
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
5.Exit
Note: You can use this option only before deletion of the last node of the circular doubly link
ed list.
Enter your choice2

Enter the position at which you want to insert the node (0th position to the 3th position of the
linked list)0

Enter the data4
```

```
4 1 2 3 4
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
5.Exit
Note: You can use this option only before deletion of the last node of the circular doubly link
ed list.
Enter your choice2

Enter the position at which you want to insert the node (0th position to the 4th position of the
linked list)3

Enter the data5

4 1 2 5 3 4
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
5.Exit
Note: You can use this option only before deletion of the last node of the circular doubly link
ed list.
Enter your choice2

Enter the position at which you want to insert the node (0th position to the 5th position of the
linked list)5

Enter the data6

4 1 2 5 3 6 4
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
5.Exit
Note: You can use this option only before deletion of the last node of the circular doubly link
ed list.
Enter your choice3

Enter the position at which you want to delete the node (0th position to the 5th position of the
 linked list)0

1 2 5 3 6 1
Main Menu
1.Creation of the List
2.Insert
```

```
3.Delete
4.Display
5.Exit
Note: You can use this option only before deletion of the last node of the circular doubly link
ed list.
Enter your choice3

Enter the position at which you want to delete the node (0th position to the 4th position of the
 linked list)2

1 2 3 6 1
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
5.Exit
Note: You can use this option only before deletion of the last node of the circular doubly link
ed list.
Enter your choice3

Enter the position at which you want to delete the node (0th position to the 3th position of the
 linked list)3

1 2 3 1
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
5.Exit
Note: You can use this option only before deletion of the last node of the circular doubly link
ed list.
Enter your choice4

1 2 3 1
Main Menu
1.Creation of the List
2.Insert
3.Delete
4.Display
5.Exit
Note: You can use this option only before deletion of the last node of the circular doubly link
ed list.
Enter your choice5


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

Trees

Q1) WAP to create a binary search tree and perform following operations:

(a) Insertion

(b) Searching

Ans)

Code:

```
#include<stdio.h>
#include<stdlib.h>
struct node{
int key;
struct node *left, *right;
};
struct node *newNode(int item){
struct node *temp = (struct node *)malloc(sizeof(struct node));
temp->key = item;
temp->left = temp->right = NULL;
return temp;
}
void traversetree(struct node *root){
if (root != NULL){
traversetree(root->left);
printf("%d \t", root->key);
traversetree(root->right);
}
}
struct node* search(struct node* root, int key){
if (root == NULL || root->key == key)
return root;
if (root->key < key)
return search(root->right, key);
return search(root->left, key);
}
struct node* insert(struct node* node, int key){
if (node == NULL) return newNode(key);
if (key < node->key)
node->left = insert(node->left, key);
else if (key > node->key)
node->right = insert(node->right, key);
return node;
}
int main(){
struct node *root = NULL;
root = insert(root, 23);
insert(root, 15);
insert(root, 12);
insert(root, 17);
insert(root, 32);
insert(root, 29);
insert(root, 45);
printf("The tree is :\n");
traversetree(root);
printf("\nSearching for 12 in this tree ");
if(search(root , 12))
printf("\nelement found");
else
printf("\nelement not found");
return 0;
}
```

Output:

```
The tree is :
12      15      17      23      29      32      45
```

```
Searching for 12 in this tree
element found

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

Q2) WAP to create a binary search tree and perform following operations:

(a) Insertion

(b) Deleion

Ans)

Code:

```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
struct node
{
int info;
struct node *lchild;
struct node *rchild;
}*root;
void find(int item,struct node **par,struct node **loc)
{
struct node *ptr,*ptrsave;

if(root==NULL)
{
*loc=NULL;
*par=NULL;
return;
}
if(item==root->info)
{
```

```
*loc=root;
*par=NULL;
return;
}
if(item<root->info)
ptr=root->lchild;
else
ptr=root->rchild;
ptrsave=root;

while(ptr!=NULL)
{
if(item==ptr->info)
{       *loc=ptr;
*par=ptrsave;
return;
}
ptrsave=ptr;
if(item<ptr->info)
ptr=ptr->lchild;
else
ptr=ptr->rchild;
}
*loc=NULL;
*par=ptrsave;
}
void insert(int item)
{
struct node *tmp,*parent,*location;
find(item,&parent,&location);
```

```
if(location!=NULL)
{
printf("Item already present");
return;
}

tmp=(struct node *)malloc(sizeof(struct node));
tmp->info=item;
tmp->lchild=NULL;
tmp->rchild=NULL;

if(parent==NULL)
root=tmp;
else
if(item<parent->info)
parent->lchild=tmp;
else
parent->rchild=tmp;
}
void case_a(struct node *par,struct node *loc )
{
if(par==NULL)
root=NULL;
else
if(loc==par->lchild)
par->lchild=NULL;
else
par->rchild=NULL;
}
void case_b(struct node *par,struct node *loc)
```

```
{
struct node *child;
if(loc->lchild!=NULL)
child=loc->lchild;
else
child=loc->rchild;

if(par==NULL )
root=child;
else
if( loc==par->lchild)
par->lchild=child;
else
par->rchild=child;
}
void case_c(struct node *par,struct node *loc)
{
struct node *ptr,*ptrsave,*suc,*parsuc;
ptrsave=loc;
ptr=loc->rchild;
while(ptr->lchild!=NULL)
{
ptrsave=ptr;
ptr=ptr->lchild;
}
suc=ptr;
parsuc=ptrsave;

if(suc->lchild==NULL && suc->rchild==NULL)
case_a(parsuc,suc);
```

```
else
case_b(parsuc,suc);

if(par==NULL)
root=suc;
else
if(loc==par->lchild)
par->lchild=suc;
else
par->rchild=suc;

suc->lchild=loc->lchild;
suc->rchild=loc->rchild;
}
int del(int item)
{
struct node *parent,*location;
if(root==NULL)
{
printf("Tree empty");
return 0;
}

find(item,&parent,&location);
if(location==NULL)
{
printf("Item not present in tree");
return 0;
}
```

```
if(location->lchild==NULL && location->rchild==NULL)
case_a(parent,location);
if(location->lchild!=NULL && location->rchild==NULL)
case_b(parent,location);
if(location->lchild==NULL && location->rchild!=NULL)
case_b(parent,location);
if(location->lchild!=NULL && location->rchild!=NULL)
case_c(parent,location);
free(location);
}/*End of del()*/

int preorder(struct node *ptr)
{
if(root==NULL)
{
printf("Tree is empty");
return 0;
}
if(ptr!=NULL)
{
printf("%d  ",ptr->info);
preorder(ptr->lchild);
preorder(ptr->rchild);
}
}
void inorder(struct node *ptr)
{
if(root==NULL)
{
printf("Tree is empty");
```

```
return;
}
if(ptr!=NULL)
{
inorder(ptr->lchild);
printf("%d  ",ptr->info);
inorder(ptr->rchild);
}
}
void postorder(struct node *ptr)
{
if(root==NULL)
{
printf("Tree is empty");
return;
}
if(ptr!=NULL)
{
postorder(ptr->lchild);
postorder(ptr->rchild);
printf("%d  ",ptr->info);
}
}
int main()
{
int choice,num;
root=NULL;
while(1)
{
printf("\n");
printf("1.Insert\n");
printf("2.Delete\n");
printf("3.Inorder Traversal\n");
printf("4.Preorder Traversal\n");
printf("5.Postorder Traversal\n");
printf("6.Display\n");
printf("7.Quit\n");
printf("Enter your choice : ");
scanf("%d",&choice);

switch(choice)
{
case 1:printf("Enter the number to be inserted : ");
scanf("%d",&num);
insert(num);
break;
case 2:printf("Enter the number to be deleted : ");
scanf("%d",&num);
del(num);
break;
case 3:inorder(root);
break;
case 4:preorder(root);
break;
case 5:postorder(root);
break;
case 6:exit(0);
break;
default:
printf("Wrong choice\n");
}
}
}
```

Output:

```
1.Insert
2.Delete
3.Inorder Traversal
4.Preorder Traversal
5.Postorder Traversal
6.Quit
Enter your choice : 1
Enter the number to be inserted : 34

1.Insert
2.Delete
3.Inorder Traversal
4.Preorder Traversal
5.Postorder Traversal
6.Quit
Enter your choice : 1
Enter the number to be inserted : 23

1.Insert
2.Delete
3.Inorder Traversal
4.Preorder Traversal
5.Postorder Traversal
6.Quit
Enter your choice : 1
Enter the number to be inserted : 23
Item already present
1.Insert
2.Delete
3.Inorder Traversal
4.Preorder Traversal
5.Postorder Traversal
6.Quit
Enter your choice : 1
Enter the number to be inserted : 65

1.Insert
2.Delete
3.Inorder Traversal
4.Preorder Traversal
5.Postorder Traversal
6.Quit
Enter your choice : 1
Enter the number to be inserted : 87

1.Insert
2.Delete
3.Inorder Traversal
4.Preorder Traversal
5.Postorder Traversal
6.Quit
Enter your choice : 1
Enter the number to be inserted : 34
Item already present
1.Insert
2.Delete
3.Inorder Traversal
4.Preorder Traversal
5.Postorder Traversal
6.Quit
Enter your choice : 1
Enter the number to be inserted : 83

1.Insert
2.Delete
3.Inorder Traversal
4.Preorder Traversal
5.Postorder Traversal
6.Quit
Enter your choice : 1
Enter the number to be inserted : 92

1.Insert
2.Delete
3.Inorder Traversal
4.Preorder Traversal
5.Postorder Traversal
6.Quit
Enter your choice : 2
Enter the number to be deleted : 56
Item not present in tree
1.Insert
2.Delete
3.Inorder Traversal
4.Preorder Traversal
5.Postorder Traversal
6.Quit
Enter your choice : 2
Enter the number to be deleted : 92

1.Insert
2.Delete
3.Inorder Traversal
```

```
4.Preorder Traversal
5.Postorder Traversal
6.Quit
Enter your choice : 2
Enter the number to be deleted : 23

1.Insert
2.Delete
3.Inorder Traversal
4.Preorder Traversal
5.Postorder Traversal
6.Quit
Enter your choice : 3
34  65  83  87
1.Insert
2.Delete
3.Inorder Traversal
4.Preorder Traversal
5.Postorder Traversal
6.Quit
Enter your choice : 4
34  65  87  83
1.Insert
2.Delete
3.Inorder Traversal
4.Preorder Traversal
5.Postorder Traversal
6.Quit
Enter your choice : 5
83  87  65  34
1.Insert
2.Delete
3.Inorder Traversal
4.Preorder Traversal
5.Postorder Traversal
6.Quit
Enter your choice : 6


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

Q3) WAP to create a binary tree and perform following operations:

(a) Preorder traversal

(b) Postorder traversal

(c) Inorder treversal

(d) Count no of nodes

(e) Count no of leaves

Ans)

Code:

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

struct node{
    int data;
    struct node *left;
    struct node *right;
}*root=NULL;

struct node *newNode(int num) {
    struct node *temp = (struct node *)malloc(sizeof(struct node));
    temp->data = num;
    temp->left = temp->right = NULL;
    if(root==NULL){
        root=temp;
    }
    return temp;
}

struct node *insert(int num, struct node *curr){

    if(curr==NULL)
        return newNode(num);



    if(num>curr->data)
        curr->right=insert(num, curr->right);
    else
        curr->left=insert(num, curr->left);

    return curr;
}

void preOrder(struct node *p)              //Preorder
{
    if(p)
    {
        printf("%d ",p->data);
        preOrder(p->left);
        preOrder(p->right);
    }
}

void postOrder(struct node *p)             //Postorder
{
    if(p)
    {
        postOrder(p->left);
        postOrder(p->right);
        printf("%d ",p->data);
    }
}

void inOrder(struct node *p)               //Inorder
{
    if(p)
    {
        inOrder(p->left);
        printf("%d ",p->data);
        inOrder(p->right);
    }
}

int countNodes(struct node *p){
    int count=1;
    if(p){
        count+=countNodes(p->left);
        count+=countNodes(p->right);
        return count;
    }
    else
        return 0;
}

int countLeaf(struct node *p){
    if(p==NULL)
        return 0;
    if(p->left==NULL && p->right==NULL)
        return 1;
    else
        return countLeaf(p->left)+countLeaf(p->right);
}

int main(){
    int i,num,count;
    do{
```

```
printf("List of Operations\n");
printf("===============\n");
printf("1.Insert\n");
printf("2.Preorder\n");
printf("3.Postorder\n");
printf("4.Inorder\n");
printf("5.Number of nodes\n");
printf("6.Number of leaves\n");
printf("Press 0 to exit\n");
printf("Enter your choice : ");
scanf("%d",&i);
if(i<=0){
   exit(1);
} else {
   switch(i)
   {
      case 1:
         printf("Enter the number to insert : ");
         scanf("%d",&num);
         insert(num,root);
         break;

      case 2:
         printf("Element(s) in the list are : ");
         preOrder(root);
         printf("\n");
         break;

      case 3:
         printf("Element(s) in the list are : ");
```

```
            postOrder(root);
            printf("\n");
            break;

          case 4:
            printf("Element(s) in the list are : ");
            inOrder(root);
            printf("\n");
            break;

          case 5:
            count=countNodes(root);
            printf("The number of nodes is %d",count);
            printf("\n");
            break;

          case 6:
            count=countLeaf(root);
            printf("The number of nodes is %d",count);
            printf("\n");
            break;

          default: printf("Invalid option\n");
        }
    }
    printf("Press 1 to continue...");
    scanf("%d",&num);
    printf("\n");
  }while(num==1);
  return 0;
```

}

Output:

```
List of Operations
===============
1.Insert
2.Preorder
3.Postorder
4.Inorder
5.Number of nodes
6.Number of leaves
Press 0 to exit
Enter your choice : 1
Enter the number to insert : 45
Press 1 to continue...1

List of Operations
===============
1.Insert
2.Preorder
3.Postorder
4.Inorder
5.Number of nodes
6.Number of leaves
Press 0 to exit
Enter your choice : 1
Enter the number to insert : 76
Press 1 to continue...1

List of Operations
===============
1.Insert
2.Preorder
3.Postorder
4.Inorder
5.Number of nodes
6.Number of leaves
Press 0 to exit
Enter your choice : 1
Enter the number to insert : 98
Press 1 to continue...1

List of Operations
===============
1.Insert
2.Preorder
3.Postorder
4.Inorder
5.Number of nodes
6.Number of leaves
```

```
Press 0 to exit
Enter your choice : 1
Enter the number to insert : 25
Press 1 to continue...1

List of Operations
===============
1.Insert
2.Preorder
3.Postorder
4.Inorder
5.Number of nodes
6.Number of leaves
Press 0 to exit
Enter your choice : 1
Enter the number to insert : 10
Press 1 to continue...1

List of Operations
===============
1.Insert
2.Preorder
3.Postorder
4.Inorder
5.Number of nodes
6.Number of leaves
Press 0 to exit
Enter your choice : 1
Enter the number to insert : 93
Press 1 to continue...1

List of Operations
===============
1.Insert
2.Preorder
3.Postorder
4.Inorder
5.Number of nodes
6.Number of leaves
Press 0 to exit
Enter your choice : 2
Element(s) in the list are : 45 25 10 76 98 93
Press 1 to continue...1

List of Operations
===============
1.Insert
2.Preorder
3.Postorder
4.Inorder
```

```
5.Number of nodes
6.Number of leaves
Press 0 to exit
Enter your choice : 3
Element(s) in the list are : 10 25 93 98 76 45
Press 1 to continue...1

List of Operations
===============
1.Insert
2.Preorder
3.Postorder
4.Inorder
5.Number of nodes
6.Number of leaves
Press 0 to exit
Enter your choice : 4
Element(s) in the list are : 10 25 45 76 93 98
Press 1 to continue...1

List of Operations
===============
1.Insert
2.Preorder
3.Postorder
4.Inorder
5.Number of nodes
6.Number of leaves
Press 0 to exit
Enter your choice : 5
The number of nodes is 6
Press 1 to continue...1

List of Operations
===============
1.Insert
2.Preorder
3.Postorder
4.Inorder
5.Number of nodes
6.Number of leaves
Press 0 to exit
Enter your choice : 6
The number of nodes is 2
Press 1 to continue...0



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

Searching and Sorting Algorithms

Q1) WAP to perform bubble sort.

Ans)

Code:

```
#include <stdio.h>
void bubblesort(int arr[100],int size)
{
for(int i=0;i<size-1;i++)
{
for(int j=0;j<size-1-i;j++)
{
if(arr[j]>arr[j+1])
{
int temp=arr[j];
arr[j]=arr[j+1];
arr[j+1]=temp;
}
}
}
for(int i=0;i<size;i++)
{
printf("%d ",arr[i]);
}
}
int main()
{
int size, arr[100];
printf("\nEnter the array limit/size: ");
scanf("%d",&size);
printf("\nEnter the array elements: ");
for(int i=0;i<size;i++)
{
scanf("%d",&arr[i]);
}
printf("The sorted array is: ");
bubblesort(arr,size);
return 0;
}
```

Output:

```
Enter the array limit/size: 10


Enter the array elements: 23 75 81 34 92 34 34 10 90 98
The sorted array is: 10 23 34 34 34 75 81 90 92 98


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

Q2) WAP to perform selection sort.

Ans)

Code:

```
#include<stdio.h>
void selectionsort(int arr[100],int size)
{
int small,pos;
for(int i=0;i<size-1;i++)
{
small=arr[i];
pos=i;
for(int j=i;j<size;j++)
{
```

```
if(small>arr[j])
{
small=arr[j];
pos=j;
}
}
int temp=arr[i];
arr[i]=arr[pos];
arr[pos]=temp;
}
for(int i=0;i<size;i++)
{
printf("%d ",arr[i]);
}
}
int main()
{
int size, arr[100];
printf("\nEnter the array limit/size: ");
scanf("%d",&size);
printf("\nEnter the array elements: ");
for(int i=0;i<size;i++)
{
scanf("%d",&arr[i]);
}
printf("The sorted array is: ");
selectionsort(arr,size);
return 0;
}
```

Output:

```
Enter the array limit/size: 10

Enter the array elements: 8 38 29 73 51 0 34 2 34 98
The sorted array is: 0 2 8 29 34 34 38 51 73 98

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

Q3) WAP to implement insertion sort.

Ans)

Code:

```
#include<stdio.h>

void insertionsort(int arr[100],int size)

{

int temp;

for(int i=0;i<size;i++)

{

temp=arr[i];

int j=i-1;

while(j>=0)

{

if(temp<arr[j])

{

arr[j+1]=arr[j];

}

else

{

break;

}

j--;

}

arr[j+1]=temp;
}
for(int i=0;i<size;i++)
{
printf("%d ",arr[i]);
}
}
int main()
{
int size, arr[100];
printf("\nEnter the array limit/size: ");
scanf("%d",&size);
printf("\nEnter the array elements: ");
for(int i=0;i<size;i++)
{
scanf("%d",&arr[i]);
}
printf("The sorted array is: ");
insertionsort(arr,size);
return 0;
}
```

Output:

```
Enter the array limit/size: 10

Enter the array elements: 64 93 24 58 23 10 2 5 29 18
The sorted array is: 2 5 10 18 23 24 29 58 64 93

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

Q4) WAP to perform shell sort.

Ans)

Code:

```
include<stdio.h>
```

```
void shellsort(int arr[100],int size)
{
for(int gap=size/2;gap>=1;gap/=2)
{
for(int j=gap;j<size;j++)
{
for(int i=j-gap;i>=0;i-=gap)
{
if(arr[i+gap]>arr[i])
{
break;
}
else
{
int temp=arr[i];
arr[i]=arr[i+gap];
arr[i+gap]=temp;
}
}
}
}
for(int i=0;i<size;i++)
{
printf("%d ",arr[i]);
}
}
int main()
{
int size, arr[100];
printf("\nEnter the array limit/size: ");
scanf("%d",&size);
printf("\nEnter the array elements: ");
for(int i=0;i<size;i++)
{
scanf("%d",&arr[i]);
}
printf("The sorted array is: ");
shellsort(arr,size);
return 0;
}
```

Output:

```
Enter the array limit/size: 10

Enter the array elements: 934 23 82 134 45 0 9 23 1 45
The sorted array is: 0 1 9 23 23 45 45 82 134 934

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

Q5) WAP to create a max-heap and perform various operations.

Ans)

Code:

```
#include <stdio.h>
int main()
{
int arr[10], no,  i, j, c, heap_root, temp;
printf("Input number of elements: ");
scanf("%d", &no);
printf("\nInput array values one by one : ");
for (i = 0; i <  no; i++)
scanf("%d", &arr[i]);
for (i = 1; i <  no; i++)
{
```

```
c = i;
do
{
heap_root  = (c - 1) / 2;
/* to create MAX arr  array */
if  (arr[heap_root] < arr[c])
{
temp =  arr[heap_root];
arr[heap_root] = arr[c];
arr[c]  = temp;
}
c =  heap_root;
} while (c !=  0);
}

printf("Heap  array : ");
for (i = 0; i <  no; i++)
printf("%d\t ", arr[i]);
for (j = no - 1; j  >= 0; j--)
{
temp = arr[0];
arr[0] = arr[j];
arr[j] = temp;
heap_root = 0;
do
{
c = 2 *  heap_root + 1;
if  ((arr[c] < arr[c + 1]) && c < j-1)
c++;
if  (arr[heap_root]<arr[c] && c<j)
{
temp =  arr[heap_root];
arr[heap_root] = arr[c];
arr[c]  = temp;
}
heap_root  = c;
} while (c  < j);
}
printf("\nSorted  array : ");
for (i = 0; i <  no; i++)
printf("\t%d", arr[i]);
printf("\n");
}
```

Output:

```
Input number of elements: 5

Input array values one by one : 23 45 17 29 49
Heap  array : 49        45      17      23      29
Sorted  array :         17      23      29      45      49


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

Q6) WAP to implement heap sort using heapify method.

Ans)

Code:

```
#include<stdio.h>
void heapsort(int[],int);
void heapify(int[],int);
void adjust(int[],int);
int main()
{
int n,i,a[50];
```

```
printf("Enter the limit: ");
scanf("%d",&n);
printf("Enter the elements: ");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
heapsort(a,n);
printf("\nThe Sorted Elements Are: ");
for(i=0;i<n;i++)
printf("%d ",a[i]);
printf("\n");
}
void heapsort(int a[],int n)
{
int i,t;
heapify(a,n);
for(i=n-1;i>0;i--)
{
t = a[0];
a[0] = a[i];
a[i] = t;
adjust(a,i);
}
}
void heapify(int a[],int n)
{
int k,i,j,item;
for(k=1;k<n;k++)
{
item = a[k];
i = k;
j = (i-1)/2;
while((i>0)&&(item>a[j]))
{
a[i] = a[j];
i = j;
j = (i-1)/2;
}
a[i] = item;
}
}
void adjust(int a[],int n)
{
int i,j,item;
j = 0;
item = a[j];
i = 2*j+1;
while(i<=n-1)
{
if(i+1 <= n-1)
if(a[i] <a[i+1])
i++;
if(item<a[i])
{
a[j] = a[i];
j = i;
i = 2*j+1;
}
else
break;
}
```

```
a[j] = item;
}
```

Output:

```
Enter the limit: 6
Enter the elements: 45 37 21 94 40 39

The Sorted Elements Are: 21 37 39 40 45 94


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

Q7) WAP to implement merge sort.

Ans)

Code:

```
#include<stdio.h>
void merge(int arr[],int si,int mid,int ei)
{
int b[100],c[100],d[100];
int m=0,n=0;
for(int i=si;i<=mid;i++)
{
c[m]=arr[i];
m++;
}
for(int i=mid+1;i<=ei;i++)
{
d[n]=arr[i];
n++;
}
int i=0;
int j=0;
int k=0;
while(i<m && j<n)
```

```
{
if(c[i]<=d[j])
{
b[k]=c[i];
i++;
k++;
}
else
{
b[k]=d[j];
j++;
k++;
}
}
if(i==m)
{
while(j<n)
{
b[k]=d[j];
j++;
k++;
}
}
else
{
while(i<m)
{
b[k]=c[i];
i++;
k++;
}
}
int l=0;
for(int i=si;i<=ei;i++)
{
arr[i]=b[l];
l++;
}
}
void mergesort(int arr[],int si,int ei)
{
if(si>=ei)
{
return;
}
int mid=(si+ei)/2;
mergesort(arr,si,mid);
mergesort(arr,mid+1,ei);
merge(arr,si,mid,ei);
return;
}
int main()
{
int size, arr[100];
printf("\nEnter the array limit/size: ");
scanf("%d",&size);
printf("\nEnter the array elements: ");
for(int i=0;i<size;i++)
{
scanf("%d",&arr[i]);
```

```
}
printf("The sorted array is: ");
mergesort(arr,0,size-1);
for(int i=0;i<size;i++)
{
printf("%d ",arr[i]);
}
return 0;
}
```

Output:

```
Enter the array limit/size: 6

Enter the array elements: 57 34 93 20 45 29
The sorted array is: 20 29 34 45 57 93

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

Q8) WAP to perform quick sort.

Ans)

Code:

```
#include<stdio.h>
int pivot(int arr[],int si,int ei)
{
int x=arr[si];
int c=0;
for(int i=si+1;i<=ei;i++)
{
if(x>=arr[i])
{
c++;
}
}
```

```
int p=si+c;
int temp=arr[si];
arr[si]=arr[p];
arr[p]=temp;
int i=si,j=ei;
while(i<p && j>p)
{
if(arr[i]<=x)
{
i++;
}
else if(arr[j]>x)
{
j--;
}
else
{
int temp=arr[i];
arr[i]=arr[j];
arr[j]=temp;
i++;
j--;
}
}
return p;
}
void quicksort(int arr[],int si,int ei)
{
if(si>=ei)
{
```

```
return;
}
int c=pivot(arr,si,ei);
quicksort(arr,si,c-1);
quicksort(arr,c+1,ei);
}
int main()
{
int size, arr[100];
printf("\nEnter the array limit/size: ");
scanf("%d",&size);
printf("\nEnter the array elements: ");
for(int i=0;i<size;i++)
{
scanf("%d",&arr[i]);
}
printf("The sorted array is: ");
quicksort(arr,0,size-1);
for(int i=0;i<size;i++)
{
printf("%d ",arr[i]);
}
return 0;
}
```

Output:

```
Enter the array limit/size: 6

Enter the array elements: 27 45 90 34 58 56
The sorted array is: 27 34 45 56 58 90

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

Q9) WAP to implement linear search algorithm.

Ans)

Code:

```
#include<stdio.h>
int lsearch(int arr[],int size,int item)
{
for(int i=0;i<size;i++)
{
if(arr[i]==item)
{
return i;
}
}
return -1;
}
int main()
{
int size, arr[100],item;
printf("\nEnter the array limit/size: ");
scanf("%d",&size);
printf("\nEnter the array elements: ");
for(int i=0;i<size;i++)
{
scanf("%d",&arr[i]);
}
printf("\nEnter the item to be searched: ");
scanf("%d",&item);
int l=lsearch(arr,size,item);
if(l==-1)
{
```

```
printf("\nThe item is not found in the given array.");
}
else
{
printf("\nThe item is found in the given array at index no %d.",l);
}
return 0;
}
```

Output:

```
Enter the array limit/size: 6

Enter the array elements: 23 87 95 47 50 28

Enter the item to be searched: 47

The item is found in the given array at index no 3.

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

Q10) WAP to implement binary search algorithm.

Ans)

Code:

```
#include<stdio.h>
int lsearch(int arr[],int size,int item)
{
int beg=0,last=size-1;
int mid;
while(beg<=last)
{
mid=(beg+last)/2;
if(arr[mid]==item)
{
return mid;
}
else if(arr[mid]>item)
{
last=mid-1;
}
else
{
beg=mid+1;
}
}
return -1;
}
int main()
{
int size, arr[100],item;
printf("\nEnter the array limit/size: ");
scanf("%d",&size);
printf("\nEnter the array elements (the sorted array elements): ");
for(int i=0;i<size;i++)
{
scanf("%d",&arr[i]);
}
printf("\nEnter the item to be searched: ");
scanf("%d",&item);
int b=lsearch(arr,size,item);
if(b==-1)
{
printf("\nThe item is not found in the given array.");
}
else
```

```
{
printf("\nThe item is found in the given array at index no %d.",b);
}
return 0;
}
```

Output:

```
Enter the array limit/size: 6

Enter the array elements (the sorted array elements): 19 26 39 45 67 70

Enter the item to be searched: 26

The item is found in the given array at index no 1.

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

Graphs

Q1) Write a c program to implement breadth first traversal.

Ans)

Code:

```
#include<stdio.h>

#include<stdlib.h>

#define MAX 100

#define initial 1

#define waiting 2

#define visited 3

int n;

int adj[MAX][MAX];

int state[MAX];

void create_graph();

void BF_Traversal();

void BFS(int v);

int queue[MAX], front = -1,rear = -1;

void insert_queue(int vertex);

int delete_queue();

int isEmpty_queue();

int main()

{

create_graph();

BF_Traversal();

return 0;

}

void BF_Traversal()

{

int v;
```

```
for(v=0; v<n; v++)
state[v] = initial;
printf("Enter Start Vertex for BFS: \n");
scanf("%d", &v);
BFS(v);
}
void BFS(int v)
{
int i;
insert_queue(v);
state[v] = waiting;
while(!isEmpty_queue())
{
v = delete_queue( );
printf("%d ",v);
state[v] = visited;
for(i=0; i<n; i++)
{
if(adj[v][i] == 1 && state[i] == initial)
{
insert_queue(i);
state[i] = waiting;
}
}
}
printf("\n");
}
void insert_queue(int vertex)
{
if(rear == MAX-1)
```

```
printf("Queue Overflow\n");
else
{
if(front == -1)
front = 0;
rear = rear+1;
queue[rear] = vertex ;
}
}
int isEmpty_queue()
{
if(front == -1 || front > rear)
return 1;
else
return 0;
}
int delete_queue()
{
int delete_item;
if(front == -1 || front > rear)
{
printf("Queue Underflow\n");
exit(1);
}
delete_item = queue[front];
front = front+1;
return delete_item;
}
void create_graph()
{
```

```
int count,max_edge,origin,destin;
printf("Enter number of vertices : ");
scanf("%d",&n);
max_edge = n*(n-1);
for(count=1; count<=max_edge; count++)
{
printf("Enter edge %d( -1 -1 to quit ) : ",count);
scanf("%d %d",&origin,&destin);
if((origin == -1) && (destin == -1))
break;
if(origin>=n || destin>=n || origin<0 || destin<0)
{
printf("Invalid edge!\n");
count--;
}
else
{
adj[origin][destin] = 1;
}
}
}
```

Output:

```
Enter number of vertices : 9
Enter edge 1( -1 -1 to quit ) : 0
1
Enter edge 2( -1 -1 to quit ) : 0
3
Enter edge 3( -1 -1 to quit ) : 0
4
Enter edge 4( -1 -1 to quit ) : 1
2
Enter edge 5( -1 -1 to quit ) : 3
6
Enter edge 6( -1 -1 to quit ) : 4
7
Enter edge 7( -1 -1 to quit ) : 6
```

```
4
Enter edge 8( -1 -1 to quit ) : 6
7
Enter edge 9( -1 -1 to quit ) : 2
5
Enter edge 10( -1 -1 to quit ) : 4
5
Enter edge 11( -1 -1 to quit ) : 7
5
Enter edge 12( -1 -1 to quit ) : 7
8
Enter edge 13( -1 -1 to quit ) : -1
-1
Enter Start Vertex for BFS:
0
0 1 3 4 2 6 5 7 8


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

Q2) Write a c program to implement depth first traversal.

Ans)

Code:

```
#include<stdio.h>

void DFS(int);

int G[10][10],visited[10],n;    //n is no of vertices and graph is sorted in array G[10][10]

void main()

{

int i,j;

printf("Enter number of vertices:");

scanf("%d",&n);

//read the adjecency matrix

printf("\nEnter adjecency matrix of the graph:");

for(i=0;i<n;i++)

for(j=0;j<n;j++)

scanf("%d",&G[i][j]);

//visited is initialized to zero
for(i=0;i<n;i++)
visited[i]=0;
DFS(0);
}
void DFS(int i)
{
int j;
printf("\n%d",i);
visited[i]=1;
for(j=0;j<n;j++)
if(!visited[j]&&G[i][j]==1)
DFS(j);
}
```

Output:

```
Enter number of vertices:8

Enter adjecency matrix of the graph: 0 1 1 1 1 0 0 0
                                     1 0 0 0 0 1 0 0
                                     1 0 0 0 0 1 0 0
                                     1 0 0 0 0 0 1 0
                                     0 1 1 0 0 0 0 1
                                     0 0 0 1 1 0 0 1
                                     0 0 0 0 0 1 1 0
                                     1 0 0 0 0 0 1 0

0
1
5
3
6
4
2
7

...Program finished with exit code 8
Press ENTER to exit console.
```

Q3) Write a c program to find a minimum spanning tree using prims algorithm.

Ans)

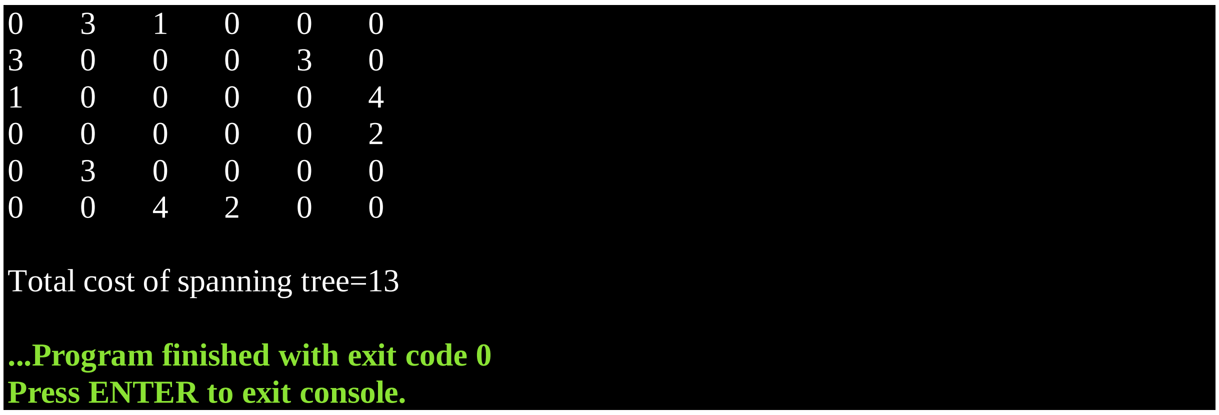
Code:

```
#include<stdio.h>
#include<stdlib.h>
#define infinity 9999
#define MAX 20
int G[MAX][MAX],spanning[MAX][MAX],n;
int prims();
int main()
{
int i,j,total_cost;
printf("Enter no. of vertices:");
scanf("%d",&n);
printf("\nEnter the adjacency matrix:\n");
for(i=0;i<n;i++)
for(j=0;j<n;j++)
scanf("%d",&G[i][j]);
total_cost=prims();
printf("\nspanning tree matrix:\n");
for(i=0;i<n;i++)
{
printf("\n");
for(j=0;j<n;j++)
printf("%d\t",spanning[i][j]);
}
printf("\n\nTotal cost of spanning tree=%d",total_cost);
return 0;
}
int prims()
```

```
{
int cost[MAX][MAX];
int u,v,min_distance,distance[MAX],from[MAX];
int visited[MAX],no_of_edges,i,min_cost,j;
//create cost[][] matrix,spanning[][]
for(i=0;i<n;i++)
for(j=0;j<n;j++)
{
if(G[i][j]==0)
cost[i][j]=infinity;
else
cost[i][j]=G[i][j];
spanning[i][j]=0;
}
//initialise visited[],distance[] and from[]
distance[0]=0;
visited[0]=1;
for(i=1;i<n;i++)
{
distance[i]=cost[0][i];
from[i]=0;
visited[i]=0;
}
min_cost=0;             //cost of spanning tree
no_of_edges=n-1;             //no. of edges to be added
while(no_of_edges>0)
{
//find the vertex at minimum distance from the tree
min_distance=infinity;
for(i=1;i<n;i++)
if(visited[i]==0&&distance[i]<min_distance)

{

v=i;

min_distance=distance[i];

}

u=from[v];

//insert the edge in spanning tree

spanning[u][v]=distance[v];

spanning[v][u]=distance[v];

no_of_edges--;

visited[v]=1;

//updated the distance[] array

for(i=1;i<n;i++)

if(visited[i]==0&&cost[i][v]<distance[i])

{

distance[i]=cost[i][v];

from[i]=v;

}

min_cost=min_cost+cost[u][v];

}

return(min_cost);

}
```

Output:

```
Enter no. of vertices: 6

Enter the adjacency matrix:
0 3 1 6 0 0
3 0 5 0 3 0
1 5 0 5 6 4
6 0 5 0 0 2
0 3 6 0 0 6
0 0 4 2 6 0

spanning tree matrix:
```



```
0	3	1	0	0	0
3	0	0	0	3	0
1	0	0	0	0	4
0	0	0	0	0	2
0	3	0	0	0	0
0	0	4	2	0	0

Total cost of spanning tree=13

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

Q4) Write a c program to find a minimum spanning tree using kruskals algorithm.

Ans)

Code:

```
#include<stdio.h>

#define MAX 30

typedef struct edge

{

int u,v,w;

}edge;

typedef struct edgelist

{

edge data[MAX];

int n;

}edgelist;

edgelist elist;

int G[MAX][MAX],n;

edgelist spanlist;

void kruskal();

int find(int belongs[],int vertexno);

void union1(int belongs[],int c1,int c2);

void sort();

void print();
void main()

{

int i,j,total_cost;

printf("\nEnter number of vertices:");

scanf("%d",&n);

printf("\nEnter the adjacency matrix:\n");

for(i=0;i<n;i++)

for(j=0;j<n;j++)

scanf("%d",&G[i][j]);

kruskal();

print();

}

void kruskal()

{

int belongs[MAX],i,j,cno1,cno2;

elist.n=0;

for(i=1;i<n;i++)

for(j=0;j<i;j++)

{

if(G[i][j]!=0)

{

elist.data[elist.n].u=i;

elist.data[elist.n].v=j;

elist.data[elist.n].w=G[i][j];

elist.n++;

}

}

sort();

for(i=0;i<n;i++)

belongs[i]=i;
```

```
spanlist.n=0;
for(i=0;i<elist.n;i++)
{
cno1=find(belongs,elist.data[i].u);
cno2=find(belongs,elist.data[i].v);
if(cno1!=cno2)
{
spanlist.data[spanlist.n]=elist.data[i];
spanlist.n=spanlist.n+1;
union1(belongs,cno1,cno2);
}
}
}
int find(int belongs[],int vertexno)
{
return(belongs[vertexno]);
}
void union1(int belongs[],int c1,int c2)
{
int i;
for(i=0;i<n;i++)
if(belongs[i]==c2)
belongs[i]=c1;
}
void sort()
{
int i,j;
edge temp;
```

```
for(i=1;i<elist.n;i++)
for(j=0;j<elist.n-1;j++)
if(elist.data[j].w>elist.data[j+1].w)
{
temp=elist.data[j];
elist.data[j]=elist.data[j+1];
elist.data[j+1]=temp;
}
}
void print()
{
int i,cost=0;
for(i=0;i<spanlist.n;i++)
{
printf("\n%d\t%d\t%d",spanlist.data[i].u,spanlist.data[i].v,spanlist.data[i].w);
cost=cost+spanlist.data[i].w;
}
printf("\n\nCost of the spanning tree=%d",cost);
}
```

Output:

```
Enter number of vertices:6

Enter the adjacency matrix:
0 3 1 6 0 0
3 0 5 0 3 0
1 5 0 5 6 4
6 0 5 0 0 2
0 3 6 0 0 6
0 0 4 2 6 0

2       0       1
5       3       2
1       0       3
4       1       3
5       2       4

Cost of the spanning tree=13
```

```
...Program finished with exit code 30
Press ENTER to exit console.
```