1) Write a program to simulate the working of stack using an array with the following :

a) Push b) Pop c) Display

The program should print appropriate messages for stack overflow, stack underflow

```
#include<stdio.h>
#include<process.h>
#include<stdlib.h>
#define MAX 5
int top=-1,stack[MAX];
void push();
void pop();
void display();
int main()
{       int ch=0;
        while(ch!=4)
        {
                printf("\n*** Stack Menu ***");
                printf("\n\n1.Push\n2.Pop\n3.Display\n4.Exit");
                printf("\n\nEnter your choice(1-4):");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1: push();
                                        break;
                        case 2: pop();
                                        break;
                        case 3: display();
                                        break;
                        case 4: exit(0);
                        default: printf("\nWrong Choice!!");
                }
        }
```

```
	return 0;
}
void push()
{
	int val;
	if(top==MAX-1)
	{
		printf("\nStack Overflow!!");
	}
	else
	{
		printf("\nEnter element to push:");
		scanf("%d",&val);
		top=top+1;
		stack[top]=val;
	}
}
void pop()
{
	if(top==-1)
	{
		printf("\nStack Underflow!!");
	}
	else
	{
		printf("\nDeleted element is %d",stack[top]);
		top=top-1;
	} }
void display()
{
	int i;
```

```
        if(top==-1)
        {
                printf("\nStack is empty!!");
        }
        else
        {
                printf("\nStack is...\n");
                for(i=top;i>=0;--i)
                        printf("%d\n",stack[i]);
        }
}
```


C:\WINDOWS\SYSTEM32\cmd.exe
Enter element to push:13
*** Stack Menu ***
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):1
Enter element to push:14
*** Stack Menu ***
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):1
Enter element to push:15
*** Stack Menu ***
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):1
Stack Overflow!!
*** Stack Menu ***
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):3


C:\WINDOWS\SYSTEM32\cmd.exe
Enter your choice(1-4):1
Stack Overflow!!
*** Stack Menu ***
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):3
Stack is...
15
14
13
12
11
*** Stack Menu ***
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):2
Deleted element is 15
*** Stack Menu ***
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):3
Stack is...
14
13
12


C:\WINDOWS\SYSTEM32\cmd.exe
Enter your choice(1-4):3
Stack is...
14
13
12
11
*** Stack Menu ***
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):2
Deleted element is 14
*** Stack Menu ***
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):3
Stack is...
13
12
11
*** Stack Menu ***
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):2
Deleted element is 13
*** Stack Menu ***

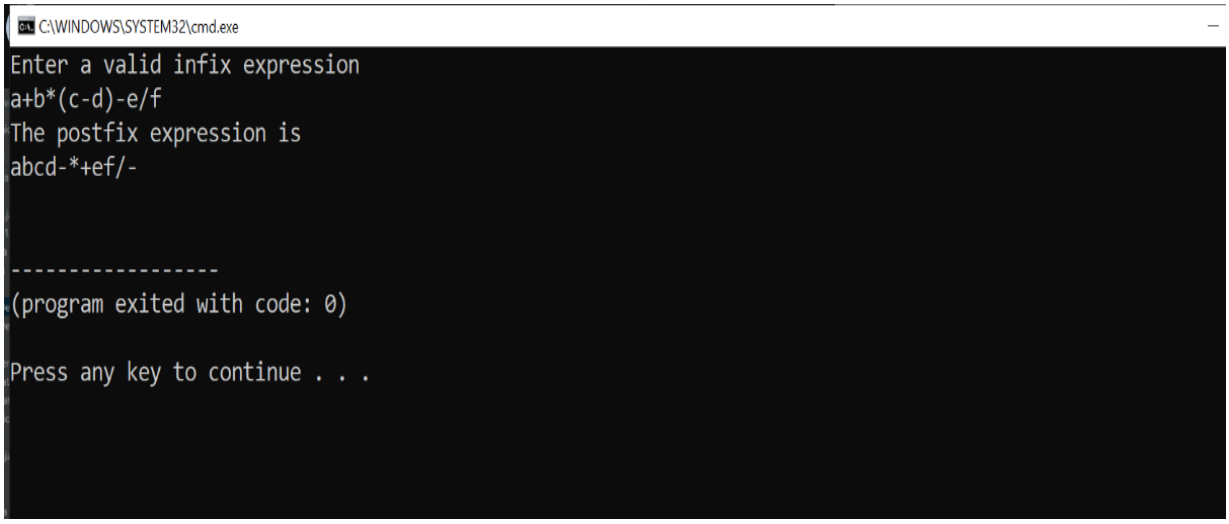

C:\WINDOWS\SYSTEM32\cmd.exe
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):2
Deleted element is 13
*** Stack Menu ***
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):3
Stack is...
12
11
*** Stack Menu ***
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):2
Deleted element is 12
*** Stack Menu ***
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):3
Stack is...
11


C:\WINDOWS\SYSTEM32\cmd.exe
11
*** Stack Menu ***
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):2
Deleted element is 11
*** Stack Menu ***
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):3
Stack is empty!!
*** Stack Menu ***
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):2
Stack Underflow!!
*** Stack Menu ***
1.Push
2.Pop
3.Display
4.Exit
Enter your choice(1-4):4
----

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

```
#include<stdio.h>
#include<process.h>
#include<string.h>
int F(char symbol){
switch(symbol){
case '+':
case'-': return 2;
case '*':
case '/': return 4;
case '^':
case '$': return 5;
case '(': return 0;
case '#': return -1;
default: return 8;
}
}
int G(char symbol){
        switch(symbol){
                case '+':
case'-': return 1;
case '*':
case '/': return 3;
case '^':
case '$': return 6;
case '(': return 9;
case ')': return 0;
default: return 7;
}
```

```
}
void infix_postfix(char infix[],char postfix[]){
        int top,i,j;
        char s[30],symbol;
        top=-1;
        s[++top]='#';
        j=0;
        for(i=0;i<strlen(infix);i++){
                symbol=infix[i];
                while(F(s[top])>G(symbol)){
                        postfix[j]=s[top--];
                        j++;
                }
        if(F(s[top])!=G(symbol))
        s[++top]=symbol;
        else
        top--;
}
while(s[top]!='#'){
postfix[j++]=s[top--];
}
postfix[j]='\0';
}
int main(){
        char infix[20],postfix[20];
        printf("Enter a valid infix expression\n");
        scanf("%s",infix);
        infix_postfix(infix,postfix);
        printf("The postfix expression is \n");
        printf("%s\n",postfix);
        return 0;  }
```



3) WAP to simulate the working of a queue of integers using an array. Provide the following operations

a) Insert b) Delete c) Display

The program should print appropriate messages for queue empty and queue overflow conditions*/

```
#include<stdio.h>
#include<process.h>
#define QUE_SIZE 3
int item,front=0,rear=-1,q[10];
void insert(){
if(rear==QUE_SIZE-1){
        printf("Queue Overflow\n");
        return;
}
rear=rear+1;
q[rear]=item;
}
int delete(){
        if(front>rear){
                front=0;
```

```
                rear=-1;
                return -1;
        }
        return q[front++];
}
void display(){
        int i;
        if(front>rear){
                printf("Queue is empty\n");
                return;
        }
        printf("Contents of queue:\n");
        for(i=front;i<=rear;i++)
        printf("%d\n",q[i]);
}
int main(){
        int count=0;
        int choice;
        for(;;){
                printf("\n1.insert\n2.delete\n3.display\n4.display size of queue\n5.exit\n");
                printf("Enter your choice\n");
                scanf("%d",&choice);
                switch(choice){
                        case 1:{
                        printf("Enter the item to be inserted\n");
                        scanf("%d",&item);
                        insert();
                        count++;
                        break;}
                        case 2:
                                item=delete();
```

```
                    if(item==-1)
                    printf("Queue is empty");
                    else
                    printf("Item deleted is %d\n",item);
                    count--;
                    break;
                case 3:display();
                break;
                case 4:
                printf("The size of the queue is %d",QUE_SIZE);
                break;
                default:
                exit(0);
            }
        }
        return 0;
}
```

```
C:\WINDOWS\SYSTEM32\cmd.exe
Enter your choice
1
Enter the item to be inserted
14
Queue Overflow

1.insert
2.delete
3.display
4.display size of queue
5.exit
Enter your choice
3
Contents of queue:
11
12
13

1.insert
2.delete
3.display
4.display size of queue
5.exit
Enter your choice
4
The size of the queue is 3
1.insert
2.delete
3.display
4.display size of queue
5.exit
Enter your choice
2
Item deleted is 11

1.insert
2.delete
3.display
4.display size of queue
5.exit
Enter your choice
3
Contents of queue:
```

```
C:\WINDOWS\SYSTEM32\cmd.exe
Enter your choice
2
Item deleted is 11

1.insert
2.delete
3.display
4.display size of queue
5.exit
Enter your choice
3
Contents of queue:
12
13

1.insert
2.delete
3.display
4.display size of queue
5.exit
Enter your choice
2
Item deleted is 12

1.insert
2.delete
3.display
4.display size of queue
5.exit
Enter your choice
3
Contents of queue:
13

1.insert
2.delete
3.display
4.display size of queue
5.exit
Enter your choice
2
Item deleted is 13
```

```
C:\WINDOWS\SYSTEM32\cmd.exe
Contents of queue:
13

1.insert
2.delete
3.display
4.display size of queue
5.exit
Enter your choice
2
Item deleted is 13

1.insert
2.delete
3.display
4.display size of queue
5.exit
Enter your choice
3
Queue is empty

1.insert
2.delete
3.display
4.display size of queue
5.exit
Enter your choice
4
The size of the queue is 3
1.insert
2.delete
3.display
4.display size of queue
5.exit
Enter your choice
5


------------------
(program exited with code: 0)

Press any key to continue . . .
```

4) WAP to simulate the working of a circular queue of integers using an array. Provide the following operations.

a) Insert b) Delete c) Display

The program should print appropriate messages for queue empty and queue overflow conditions*/

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
#define QUE_SIZE 3
int item,front=0,rear=-1,q[QUE_SIZE],count=0;
void insertrear()
{
if(count==QUE_SIZE)
{
printf("queue overflow\n");
return;
}
rear=(rear+1)%QUE_SIZE;
q[rear]=item;
count++;
}
int deletefront()
{
if(count==0) return -1;
item=q[front];
front=(front+1)%QUE_SIZE;
count=count-1;
return item;
}
void displayQ()
{
```

```
int i,f;
if(count==0)
{
printf("queue is empty\n");
return;
}
f=front;
printf("Contents of queue \n");
for(i=1;i<=count;i++)
{
printf("%d\n",q[f]);
f=(f+1)%QUE_SIZE;
}
}
int main()
{
 int choice;
 for(;;)
 {
 printf("\n1:insertrear\n2:deletefront\n3:display\n4:exit\n");
 printf("enter the choice\n");
 scanf("%d",&choice);

 switch(choice)
 {
 case 1:printf("enter the item to be inserted\n");
        scanf("%d",&item);
        insertrear();
        break;
 case 2:item=deletefront();
        if(item==-1)
```

```
            printf("queue is empty\n");
            else
            printf("item deleted =%d\n",item);
            break;
 case 3:displayQ();
            break;
 default:exit(0);
}
}
return 0;
}
```

5) WAP to Implement Singly Linked List with following operations

a) a) Create a linked list. b) Insertion of a node at first position, at any position and at end of list. c) Display the contents of the linked list.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
        int info;
        struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
        NODE x;
        x=(NODE)malloc(sizeof(struct node));
        if (x==NULL)
        {
                printf("Memory full\n");
                exit(0);
```

```
        }
        return x;
}
void freenode(NODE x)
{
        free(x);
}
NODE insert_rear(int item,NODE first)
{
        NODE temp,cur;
        temp=getnode();
        temp->info=item;
        temp->link=NULL;
        if (first==NULL)
        {
                return temp;
        }
        cur=first;
        while (cur->link!=NULL)
        {
                cur=cur->link;
        }
        cur->link=temp;
        return first;
}
NODE insert_pos(int item,int pos,NODE first)
{
        NODE temp,cur,prev;
        int count;
        temp=getnode();
        temp->info=item;
```

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

```
        return first;
}
NODE insert_front(NODE first,int item)
{
NODE temp;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
return temp;
temp->link=first;
first=temp;
return first;
}
void display(NODE first)
{
        NODE temp;
        if (first==NULL)
        {
                printf("Linked is empty cannot display items\n");
        }
        printf("The contents of the linked list are:\n");
        for (temp=first;temp!=NULL;temp=temp->link)
        {
                printf("%d\n",temp->info);
        }
}
int main()
{
        NODE first=NULL;
        int item,choice,pos;
```

```
	for (;;)
	{
		printf("1:Insert rear\n2:Insert at specified position\n3:Insert
front4:Display\n5:Exit\n");
		scanf("%d",&choice);
		switch (choice)
		{
			case 1:printf("Enter the item at the rear end:\n");
			    scanf("%d",&item);
			    first=insert_rear(item,first);
			    break;
			case 3:printf("enter the item at front-end\n");
scanf("%d",&item);
first=insert_front(first,item);
break;
			case 2:printf("Enter the item and the position:\n");
			    scanf("%d%d",&item,&pos);
			    first=insert_pos(item,pos,first);
			    break;
		   case 4:display(first);
		        break;
		   case 5:exit(0);
		        break;
		   default:printf("Please enter a valid value\n");
		}
	}
	return 0;
}
```


C:\WINDOWS\SYSTEM32\cmd.exe
1:Insert rear
2:Insert at specified position
3:Insert front4:Display
5:Exit
1
Enter the item at the rear end:
10
1:Insert rear
2:Insert at specified position
3:Insert front4:Display
5:Exit
1
Enter the item at the rear end:
20
1:Insert rear
2:Insert at specified position
3:Insert front4:Display
5:Exit
1
Enter the item at the rear end:
30
1:Insert rear
2:Insert at specified position
3:Insert front4:Display
5:Exit
1
Enter the item at the rear end:
40
1:Insert rear
2:Insert at specified position
3:Insert front4:Display
5:Exit
1
Enter the item at the rear end:
50
1:Insert rear
2:Insert at specified position
3:Insert front4:Display
5:Exit
1
Enter the item at the rear end:


C:\WINDOWS\SYSTEM32\cmd.exe
5:Exit
1
Enter the item at the rear end:
60
1:Insert rear
2:Insert at specified position
3:Insert front4:Display
5:Exit
4
The contents of the linked list are:
10
20
30
40
50
60
1:Insert rear
2:Insert at specified position
3:Insert front4:Display
5:Exit
3
enter the item at front-end
-10
1:Insert rear
2:Insert at specified position
3:Insert front4:Display
5:Exit
3
enter the item at front-end
-20
1:Insert rear
2:Insert at specified position
3:Insert front4:Display
5:Exit
4
The contents of the linked list are:
-20
-10
10
20
30


C:\WINDOWS\SYSTEM32\cmd.exe
4
The contents of the linked list are:
-20
-10
10
20
30
40
50
60
1:Insert rear
2:Insert at specified position
3:Insert front4:Display
5:Exit
2
Enter the item and the position:
0
3
1:Insert rear
2:Insert at specified position
3:Insert front4:Display
5:Exit
4
The contents of the linked list are:
-20
-10
0
10
20
30
40
50
60
1:Insert rear
2:Insert at specified position
3:Insert front4:Display
5:Exit
5
----

6) WAP to Implement Singly Linked List with following operations

a) a) Create a linked list. b) Deletion of first element, specified element and last element in the list. c) Display the contents of the linked list.*/

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
        int info;
        struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
        NODE x;
        x=(NODE)malloc(sizeof(struct node));
        if (x==NULL)
        {
                printf("Memory full\n");
                exit(0);
        }
        return x;
}
void freenode(NODE x)
{
        free(x);
}
NODE insert_rear(int item,NODE first)
{
        NODE temp,cur;
        temp=getnode();
        temp->info=item;
```

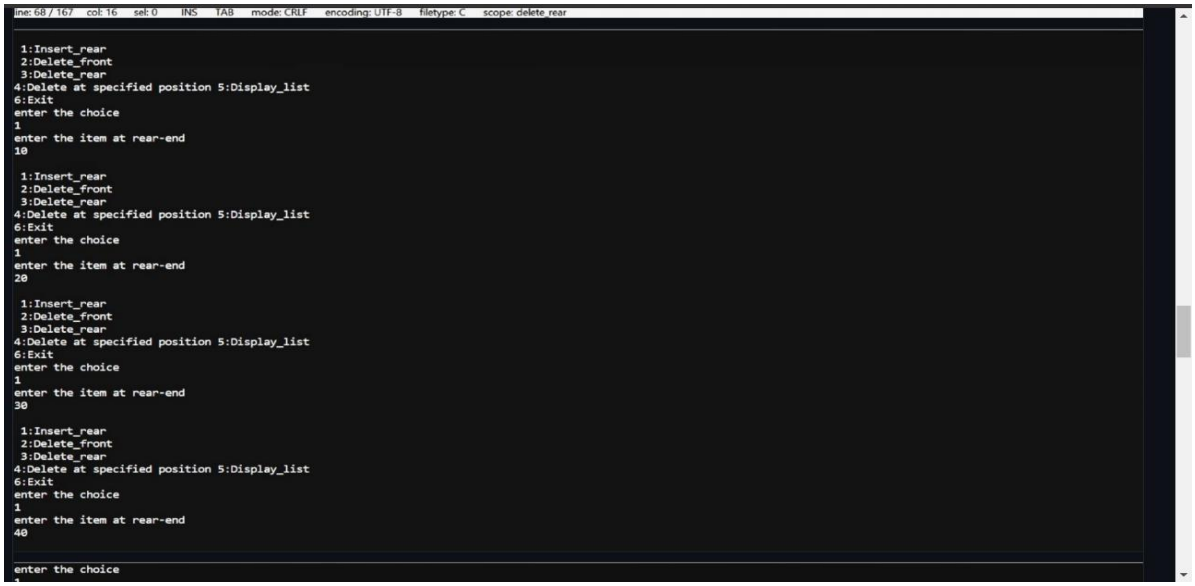
```
        temp->link=NULL;
        if (first==NULL)
        {
                return temp;
        }
        cur=first;
        while (cur->link!=NULL)
        {
                cur=cur->link;
        }
        cur->link=temp;
        return first;
}
NODE delete_rear(NODE first)
{
NODE cur,prev;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
if(first->link==NULL)
{
printf("item deleted is %d\n",first->info);
free(first);
return NULL;
}
prev=NULL;
cur=first;
while(cur->link!=NULL)
{
```

```
prev=cur;
cur=cur->link;
}
printf("iten deleted at rear-end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;
}
NODE delete_front(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
temp=first;
temp=temp->link;
printf("item deleted at front-end is=%d\n",first->info);
free(first);
return temp;
}
NODE delete_pos(int pos,NODE first)
{
        NODE prev,cur;
        int count;
        if (first==NULL || pos<=0)
        {
                printf("Invalid position\n");
                return NULL;
        }
```

```
	if (pos==1)
	{
		cur=first;
		first=first->link;
		freenode(cur);
		return first;
	}
	prev=NULL;
	cur=first;
	count=1;
	while (cur!=NULL)
	{
		if (count==pos)
		{
			break;
		}
		prev=cur;
		cur=cur->link;count++;
	}
	if (count!=pos)
	{
		printf("Invalid position\n");
		return first;
	}
	prev->link=cur->link;
	freenode(cur);
	return first;
}
void display(NODE first)
{
	NODE temp;
```

```
        if (first==NULL)
        {
                printf("Linked is empty cannot display items\n");
        }
        printf("The contents of the linked list are:\n");
        for (temp=first;temp!=NULL;temp=temp->link)
        {
                printf("%d\n",temp->info);
        }
}
int main()
{
int item,choice,pos;
NODE first=NULL;
for(;;)
{
printf("\n 1:Insert_rear\n 2:Delete_front\n 3:Delete_rear\n4:Delete at specified position
5:Display_list\n6:Exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the item at rear-end\n");
scanf("%d",&item);
first=insert_rear(item,first);
break;
case 2:first=delete_front(first);
break;
case 3:first=delete_rear(first);
break;
case 4:printf("Enter the position:\n");
```

```
                    scanf("%d",&pos);
                    first=delete_pos(pos,first);
                    break;
break;
case 5:display(first);
break;
default:exit(0);
break;
}
}
}
```



```
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
2
item deleted at front-end is=10

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
5
The contents of the linked list are:
20
30
40
50

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
3
iten deleted at rear-end is 50
 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
5
The contents of the linked list are:
20
30
40

 1:Insert_rear
 2:Delete_front
```

```
20
30
40

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
4
Enter the position:
2

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
5
The contents of the linked list are:
20
40

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
2
item deleted at front-end is=20

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
3
```

```
 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
2
item deleted at front-end is=20

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
3
item deleted is 40

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
2
list is empty cannot delete

 1:Insert_rear
 2:Delete_front
 3:Delete_rear
4:Delete at specified position 5:Display_list
6:Exit
enter the choice
6


------------------
(program exited with code: 0)

Press any key to continue . . .
```

7) WAP Implement Single Link List with following operations

a) a) Sort the linked list. b) Reverse the linked list. c) Concatenation of two linked lists

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<process.h>
struct node
{
int info;
struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{
printf("mem full\n");
exit(0);
}
return x;
}
NODE insert_rear(NODE first,int item)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
return temp;
```

```
cur=first;
while(cur->link!=NULL)
cur=cur->link;
cur->link=temp;
return first;
}
void display(NODE first)
{
NODE temp;
if(first==NULL)
printf("list empty");
for(temp=first;temp!=NULL;temp=temp->link)
{
printf("%d\n",temp->info);
}
}
NODE concat(NODE first,NODE second)
{
NODE cur;
if(first==NULL)
return second;
if(second==NULL)
return first;
cur=first;
while(cur->link!=NULL)
cur=cur->link;
cur->link=second;
return first;
}
NODE reverse(NODE first)
{
```

```
NODE cur,temp;
cur=NULL;
while(first!=NULL)
{
temp=first;
first=first->link;
temp->link=cur;
cur=temp;
}
return cur;
}
NODE order_list(int item,NODE first)
{
NODE temp,prev,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL) return temp;
if(item<first->info)
{
temp->link=first;
return temp;
}
prev=NULL;
cur=first;
while(cur!=NULL&&item>cur->info)
{
prev=cur;
cur=cur->link;
}
prev->link=temp;
```

```
temp->link=cur;
return first;
}
int main()
{
int item,choice,i,n;
NODE first=NULL,a,b;
for(;;)
{
printf("1.insert_front\n2.concat\n3.reverse\n4.order list\n5.dislay\n6.exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the item\n");
scanf("%d",&item);
first=insert_rear(first,item);
break;
case 2:printf("enter the no of nodes in 1\n");
scanf("%d",&n);
a=NULL;
for(i=0;i<n;i++)
{
printf("enter the item\n");
scanf("%d",&item);
a=insert_rear(a,item);
}
printf("enter the no of nodes in 2\n");
scanf("%d",&n);
b=NULL;
for(i=0;i<n;i++)
```

```
{
printf("enter the item\n");
scanf("%d",&item);
b=insert_rear(b,item);
}
a=concat(a,b);
display(a);
break;
case 3:first=reverse(first);
display(first);
break;
case 4:printf("enter the item to be inserted in ordered_list\n");
scanf("%d",&item);
first=order_list(item,first);
break;
case 5:display(first);
break;
default:exit(0);
}
}
}
```

```
5.dislay
6.exit
enter the choice
2
enter the no of nodes in 1
2
enter the item
11
enter the item
22
enter the no of nodes in 2
3
enter the item
33
enter the item
44
enter the item
55
11
22
33
44
55
1.insert_front
2.concat
3.reverse
4.order list
5.dislay
6.exit
enter the choice
5
10
20
1.insert_front
2.concat
3.reverse
4.order list
5.dislay
6.exit
enter the choice
1
```

```
3.reverse
4.order list
5.dislay
6.exit
enter the choice
1
enter the item
30
1.insert_front
2.concat
3.reverse
4.order list
5.dislay
6.exit
enter the choice
1
enter the item
40
1.insert_front
2.concat
3.reverse
4.order list
5.dislay
6.exit
enter the choice
5
10
20
30
40
1.insert_front
2.concat
3.reverse
4.order list
5.dislay
6.exit
enter the choice
3
40
30
20
```

```
5.dislay
6.exit
enter the choice
3
40
30
20
10
1.insert_front
2.concat
3.reverse
4.order list
5.dislay
6.exit
enter the choice
4
enter the item to be inserted in ordered_list
11
1.insert_front
2.concat
3.reverse
4.order list
5.dislay
6.exit
enter the choice
4
enter the item to be inserted in ordered_list
54
1.insert_front
2.concat
3.reverse
4.order list
5.dislay
6.exit
enter the choice
4
enter the item to be inserted in ordered_list
23
1.insert_front
2.concat
3.reverse
```

8) WAP to implement Stack & Queues using Linked Representation

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<process.h>
struct node
{
int info;
struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{
printf("mem full\n");
exit(0);
}
```

```
return x;
}
void freenode(NODE x)
{
free(x);
}
NODE insert_front(NODE first,int item)
{
NODE temp;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
return temp;
temp->link=first;
first=temp;
return first;
}
NODE delete_front(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
temp=first;
temp=temp->link;
printf("item deleted at front-end is=%d\n",first->info);
free(first);
return temp;
```

```
}
NODE insert_rear(NODE first,int item)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
return temp;
cur=first;
while(cur->link!=NULL)
cur=cur->link;
cur->link=temp;
return first;
}
NODE delete_rear(NODE first)
{
NODE cur,prev;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
if(first->link==NULL)
{
printf("item deleted is %d\n",first->info);
free(first);
return NULL;
}
prev=NULL;
cur=first;
```

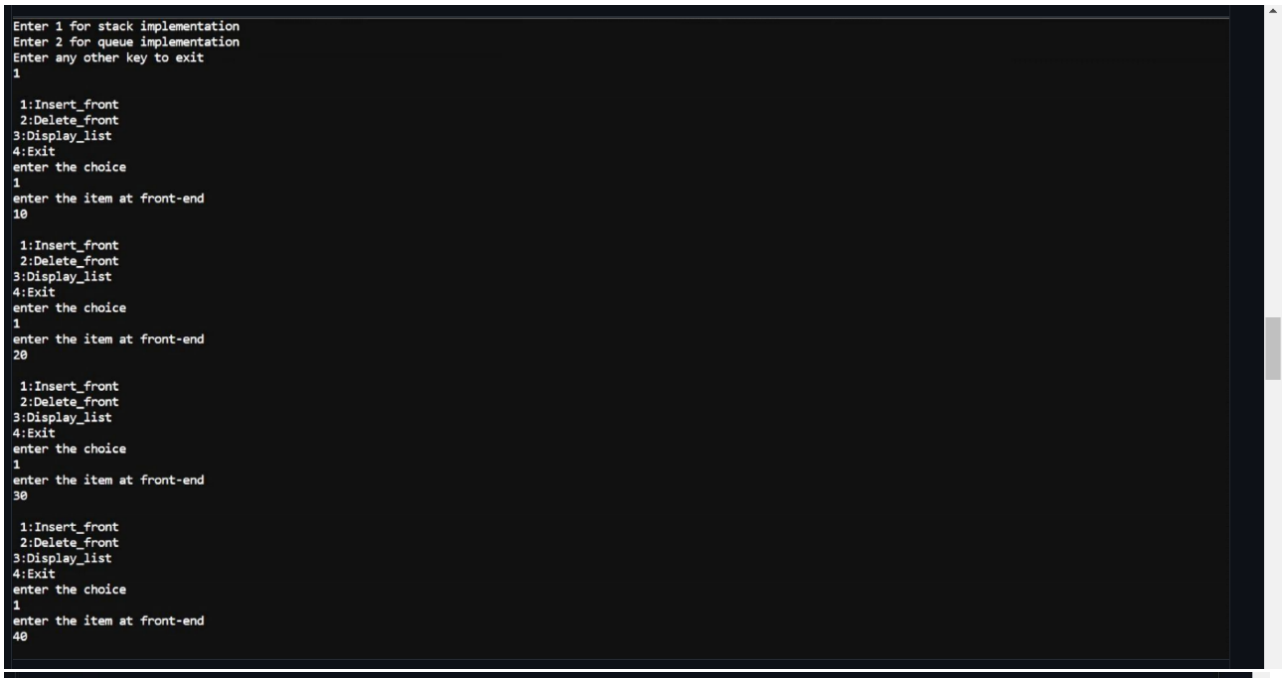
```
while(cur->link!=NULL)
{
prev=cur;
cur=cur->link;
}
printf("iten deleted at rear-end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;
}
void display(NODE first)
{
NODE temp;
if(first==NULL)
printf("list empty cannot display items\n");
for(temp=first;temp!=NULL;temp=temp->link)
{
printf("%d\n",temp->info);
}
}
int main()
{
int item,choice,choice1,choice2;
NODE first=NULL;

printf("Enter 1 for stack implementation\n");
printf("Enter 2 for queue implementation\n");
printf("Enter any other key to exit\n");
scanf("%d",&choice);
for(;;){
if(choice==1){
```

```
printf("\n 1:Insert_front\n 2:Delete_front\n3:Display_list\n4:Exit\n");
printf("enter the choice\n");
scanf("%d",&choice1);
switch(choice1)
{
case 1:printf("enter the item at front-end\n");
scanf("%d",&item);
first=insert_front(first,item);
break;
case 2:first=delete_front(first);
break;
case 3:display(first);
break;
default: exit(0);
break;
}
}
else if(choice==2){
        printf("1:Insert_rear\n2:delete_front\n3:Display_list\n4:Exit\n");
        printf("enter the choice\n");
scanf("%d",&choice2);
switch(choice2)
{
        case 1:printf("enter the item at rear-end\n");
scanf("%d",&item);
first=insert_rear(first,item);
break;
case 2:first=delete_front(first);
break;
case 3:display(first);
break;
```

```
default: exit(0);
break;
```

```
        }}
        else{
                exit(0);
        }
        }}
```



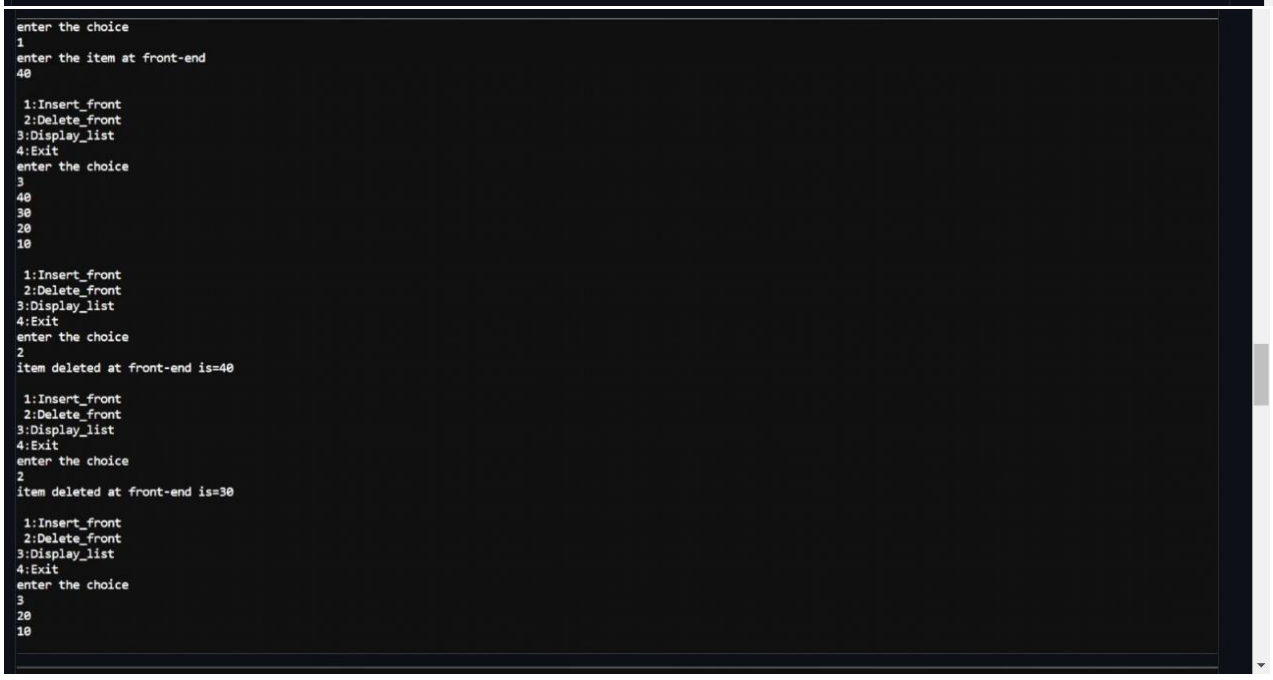
```
Enter 1 for stack implementation
Enter 2 for queue implementation
Enter any other key to exit
1

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
10

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
20

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
30

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
40
```



```
enter the choice
1
enter the item at front-end
40

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
3
40
30
20
10

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=40

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=30

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
3
20
10
```

```
 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=10

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
2
list is empty cannot delete

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
3
list empty cannot display items

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
4


------------------
(program exited with code: 0)

Press any key to continue . . .
```

```
Enter 1 for stack implementation
Enter 2 for queue implementation
Enter any other key to exit
2
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at rear-end
10
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at rear-end
20
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at rear-end
30
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at rear-end
40
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
```

```
enter the choice
1
enter the item at rear-end
40
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
3
10
20
30
40
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=10
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=20
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=30
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
3
```

```
2
```

9) WAP Implement doubly link list with primitive operations

a) a) Create a doubly linked list. b) Insert a new node to the left of the node.

b) c) Delete the node based on a specific value. c) Display the contents of the list*/

10) Write a program

a) To construct a binary Search tree.

b) To traverse the tree using all the methods i.e., in-order, preorder and post order

c) To display the elements in the tree.

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
#include<stdlib.h>
#include<string.h>
struct node
{
int info;
struct node*llink;
struct node*rlink;
};
typedef struct node*NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{
printf("memory not available");
exit(0);
}
return x;
}
void freenode(NODE x)
{
free(x);
}
```

```
NODE insert(int item,NODE root)
{
NODE temp,cur,prev;
char direction[10];
int i;
temp=getnode();
temp->info=item;
temp->llink=NULL;
temp->rlink=NULL;
if(root==NULL)
 return temp;
printf("give direction to insert\n");
scanf("%s",direction);
prev=NULL;
cur=root;
for(i=0;i<strlen(direction)&&cur!=NULL;i++)
{
prev=cur;
if(direction[i]=='l')
cur=cur->llink;
else if(direction[i]=='r')
cur=cur->rlink;
}
if(cur!=NULL||i!=strlen(direction))
{
printf("insertion not possible\n");
freenode(temp);
return(root);
}
if(cur==NULL)
{
```

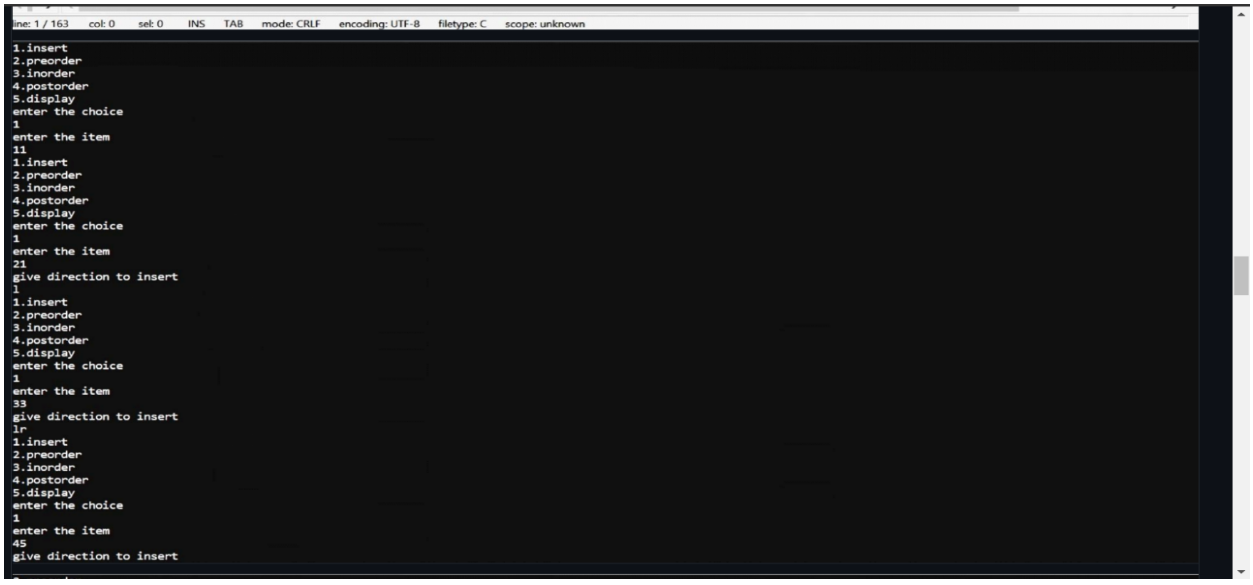
```
if(direction[i-1]=='l')
prev->llink=temp;
else
prev->rlink=temp;
}
return(root);
}
void preorder(NODE root)
{
if(root!=NULL)
{
printf("the item is %d\n",root->info);
preorder(root->llink);
preorder(root->rlink);
}
}
void inorder(NODE root)
{
if(root!=NULL)
{
inorder(root->llink);
printf("the item is%d\n",root->info);
inorder(root->rlink);
}
}
void postorder(NODE root)
{
if (root!=NULL)
{
postorder(root->llink);
postorder(root->rlink);
```

```
printf("the item is%d\n",root->info);
}
}
void display(NODE root,int i)
{
int j;
if(root!=NULL)
{
display(root->rlink,i+1);
for (j=1;j<=i;j++)
printf("  ");
printf("%d\n",root->info);
display(root->llink,i+1);
}
}

int main()
{
NODE root=NULL;
int choice,item;
for(;;)
{
printf("1.insert\n2.preorder\n3.inorder\n4.postorder\n5.display\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1: printf("enter the item\n");
                scanf("%d",&item);
                root=insert(item,root);
                break;
```

```
case 2: if(root==NULL)
                        {
                         printf("tree is empty");
                        }
                        else
                        {
                         printf("given tree is");
                         display(root,1);
                         printf("the preorder traversal is \n");
                         preorder(root);
                        }
                        break;
case 3:if(root==NULL)
                {
                        printf("tree is empty");
                }
                else
                {
                        printf("given tree is");
                        display(root,1);
                        printf("the inorder traversal is \n");
                        inorder(root);
                        }
                break;
case 4:if (root==NULL)
                        {
                        printf("tree is empty");
                         }
                else
                {
                        printf("given tree is");
```

```
                display(root,1);
                printf("the postorder traversal is \n");
                postorder(root);
            }
           break;
case 5:display(root,1);
            break;
default:exit(0);
}
}
}
```



```
1
enter the item
32
give direction to insert
r
insertion not possible
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
rl
give direction to insert
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
34
give direction to insert
rl
insertion not possible
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
67
give direction to insert
rll
1.insert
2.preorder
enter the choice
1
```

```
enter the choice
1
enter the item
67
give direction to insert
rll
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
5
    45
      32
        67
  11
        65
      33
    21
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
67
give direction to insert
ll
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
5
    45
      32
        67
11
```

```
        67
11
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
5
    45
      32
        67
  11
        65
      33
    21
      67
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
2
given tree is    45
      32
        67
  11
        65
      33
    21
      67
the preorder traversal is
the item is 11
the item is 21
the item is 67
the item is 33
the item is 65
the item is 45
the item is 32
the item is 67
1.insert
        65
```

```
       65
     33
   21
     67
the preorder traversal is
the item is 11
the item is 21
the item is 67
the item is 33
the item is 65
the item is 45
the item is 32
the item is 67
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
3
given tree is    45
      32
        67
  11
       65
     33
   21
     67
the inorder traversal is
the item is67
the item is21
the item is33
the item is65
the item is11
the item is67
the item is32
the item is45
1.insert
2.preorder
3.inorder
4.postorder
```

```
   11
        65
      33
    21
      67
  the inorder traversal is
  the item is67
  the item is21
  the item is33
  the item is65
  the item is11
  the item is67
  the item is32
  the item is45
  1.insert
  2.preorder
  3.inorder
  4.postorder
  5.display
  enter the choice
  4
  given tree is    45
        32
          67
    11
        65
      33
    21
      67
  the postorder traversal is
  the item is67
  the item is65
  the item is33
  the item is21
  the item is67
  the item is32
  the item is45
  the item is11
  1.insert
  2.preorder
  3.inorder
```

```
     33
   21
     67
the postorder traversal is
the item is67
the item is65
the item is33
the item is21
the item is67
the item is32
the item is45
the item is11
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
5
    45
      32
        67
  11
       65
     33
   21
     67
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
7


------------------
(program exited with code: 0)

Press any key to continue . . .
```