DS LAB-RECORD

Likitha B

1BM19CS079

LAB-1

WAP 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 and stack underflow.

CODE-

```
#include<stdio.h>

int stack[100],choice,n,top,x,i;

void push(void);

void pop(void);

void display(void);

int main()

{

 //clrscr();

 top=-1;

 printf("\n Enter the size of STACK[MAX=100]:");

 scanf("%d",&n);

 printf("\n\t STACK OPERATIONS USING ARRAY");

 printf("\n\t--------------------------------");

 printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");

 do

 {
```

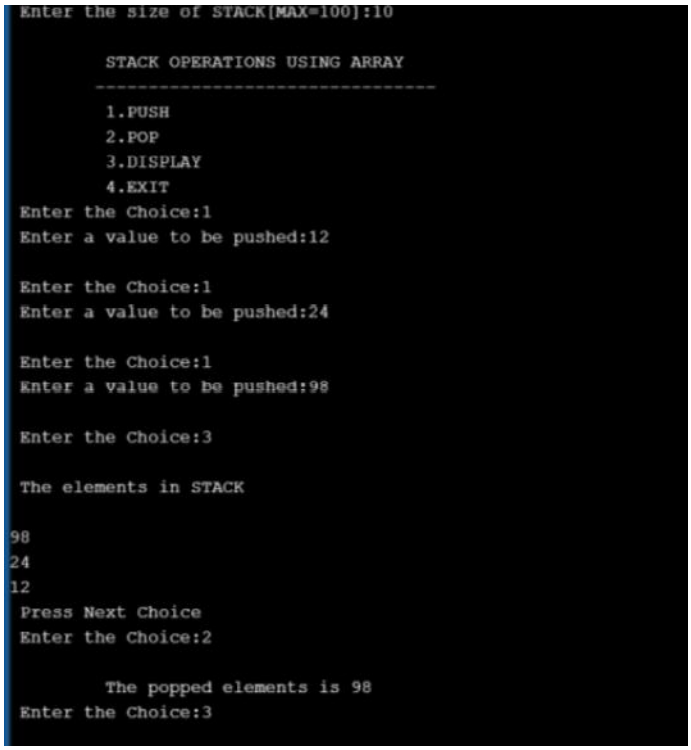
```
printf("\n Enter the Choice:");
scanf("%d",&choice);
switch(choice)
{
case 1:
{
push();
break;
}
case 2:
{
pop();
break;
}
case 3:
{
display();
break;
}
case 4:
{
printf("\n\t EXIT POINT ");
break;
}
```

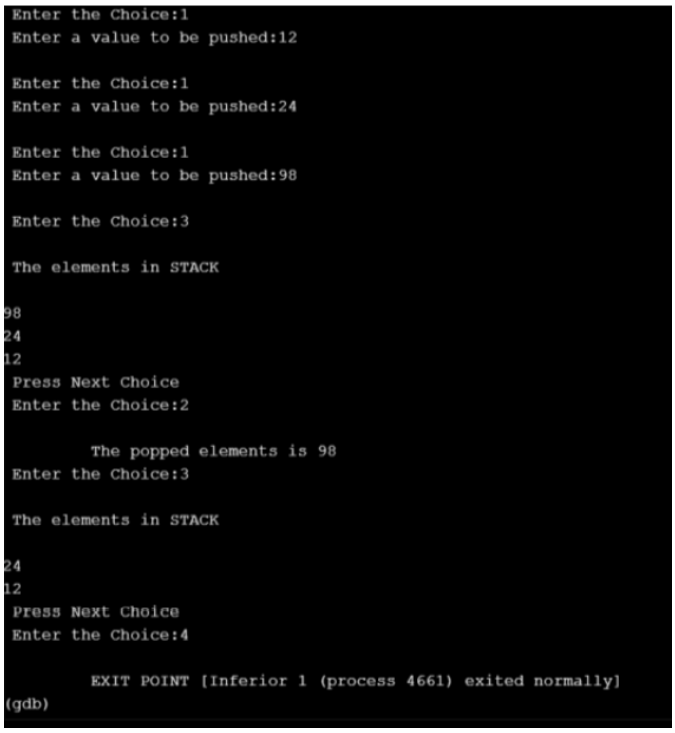
```
 default:
 {
 printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");
 }

 }
 }
 while(choice!=4);
 return 0;
}
void push()
{
 if(top>=n-1)
 {
 printf("\n\tSTACK is over flow");

 }
 else
 {
 printf(" Enter a value to be pushed:");
 scanf("%d",&x);
 top++;
 stack[top]=x;
 }
```

```
}
void pop()
{
 if(top<=-1)
 {
 printf("\n\t Stack is under flow");
 }
 else
 {
 printf("\n\t The popped elements is %d",stack[top]);
 top--;
 }
}
void display()
{
 if(top>=0)
 {
 printf("\n The elements in STACK \n");
 for(i=top; i>=0; i--)
 printf("\n%d",stack[i]);
 printf("\n Press Next Choice");
 }
 else
 {
```

```
 printf("\n The STACK is empty");

 }


}
```

OUTPUT-





LAB-2

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

CODE-

```
#include<stdio.h>

#include<string.h>

#include<stdlib.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';
}
void main()
{
 char infix[20];
 char postfix[20];
 printf("Enter the valid infix expression\n");
 scanf("%s",infix);
 infix_postfix(infix,postfix);
 printf("the postfix expression is\n");
 printf("%s\n",postfix);

}
```

OUTPUT-

LAB-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 of queue empty and queue overflow condition.

CODE-

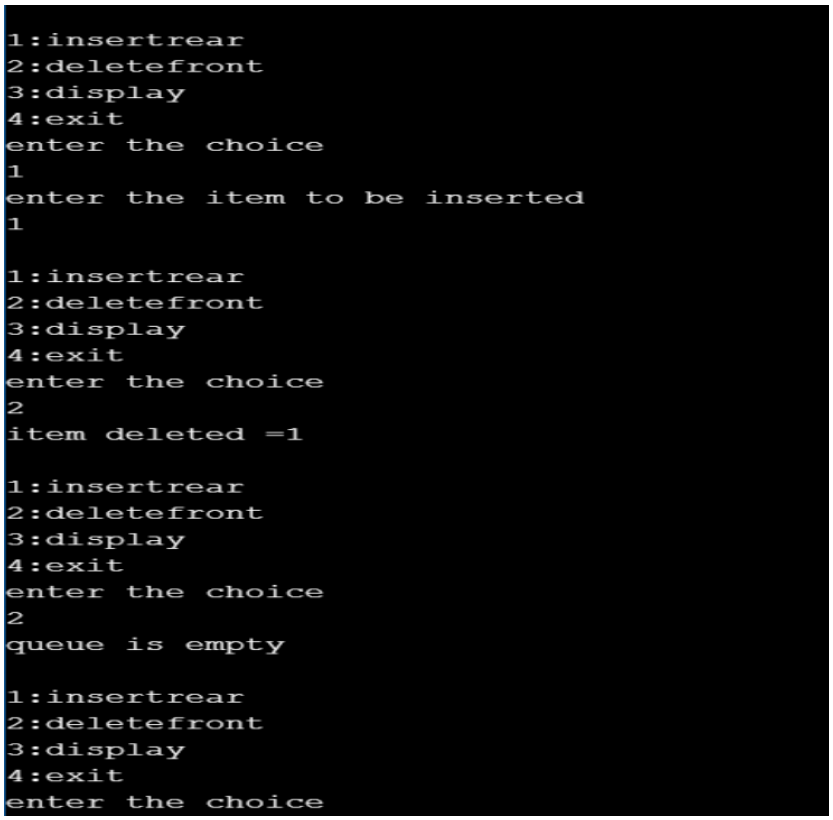
```
#include<stdio.h>

#include<conio.h>

#define QUE_SIZE 5

int item,front=0,rear=-1,q[10];

void insertrear()

{

if(rear==QUE_SIZE-1)

{

printf("queue overflow\n");

return;

}

rear=rear+1;
```

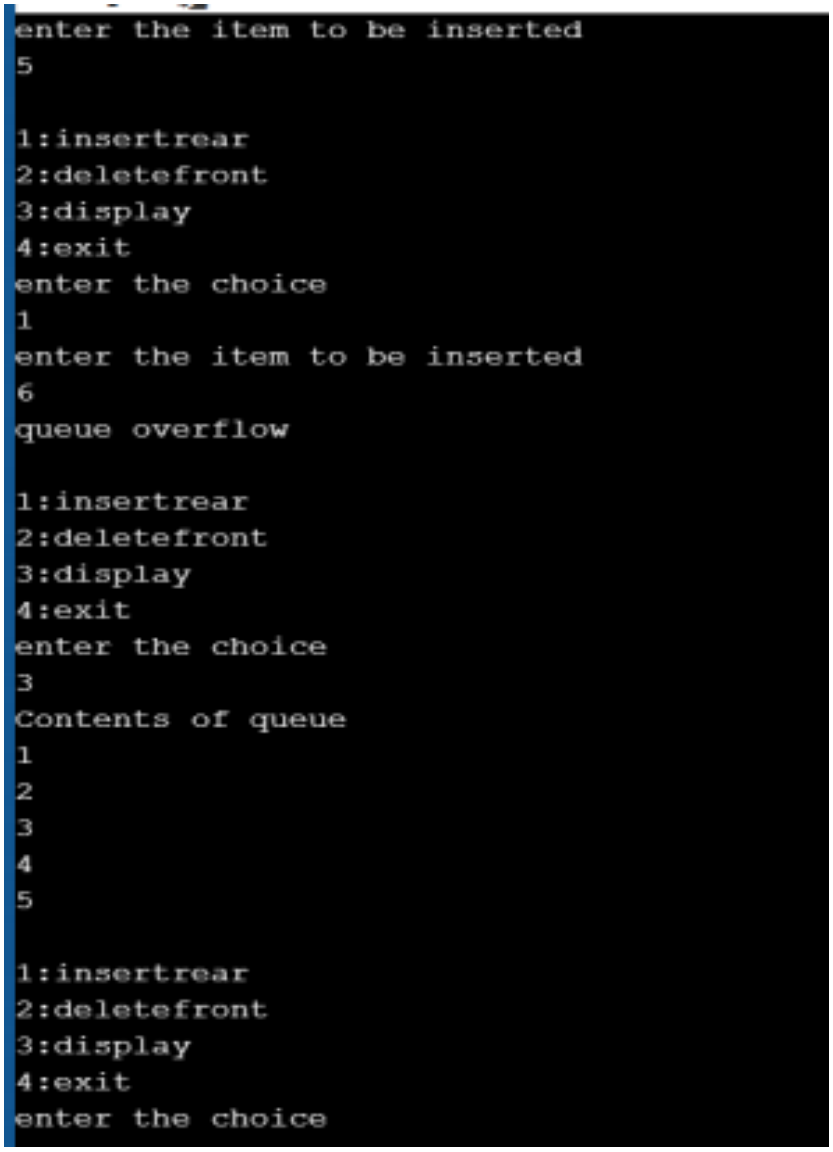
```
q[rear]=item;
}
int deletefront()
{
if(front>rear)
{
front=0;
rear=-1;
return-1;
}
return q[front++];
}
void displayQ()
{
int i;
if(front>rear)
{
printf("queue is empty\n");
return;
}
printf("Contents of queue\n");
for(i=front;i<=rear;i++)
{
printf("%d\n",q[i]);
```

```
}
}
void 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:return;

}

}

}
```

OUTPUT-





LAB-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.

CODE-

#include<stdio.h>

#include<conio.h>

```
#include<stdlib.h>
#define QUE_SIZE 5
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);
}
}
}
```

OUTPUT-

```
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
1

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
item deleted =1

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
queue is empty

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
```

```
3:display
4:exit
enter the choice
1
enter the item to be inserted
5

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
6

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
3
Contents of queue
5
6

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
```

```
3:display
4:exit
enter the choice
1
enter the item to be inserted
10

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
7

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
9

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
```

```
3:display
4:exit
enter the choice
1
enter the item to be inserted
7

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
9

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
8
queue overflow

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
```

LAB-5

WAP to implement Singly Linked List with the following operations a]create a linked list b]Insertion of node at first position and end of the list c] deletion of node at first position and end of the list d]displaying contents of the list

CODE-

#include<stdio.h>

```
#include<stdlib.h>
#include<conio.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("item 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);
 }
}
void main()
{
int item,choice,pos;
NODE first=NULL;
for(;;)
{
printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n
5:Display_list\n6:Exit\n");
printf("ENTER THE CHOICE\n");
scanf("%d",&choice);
switch(choice)
{
 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:printf("Enter the item at rear-end\n");
scanf("%d",&item);
first=insert_rear(first,item);
```

```
break;
 case 4:first=delete_rear(first);
break;
 case 5:display(first);
break;
default:exit(0);
break;
}
}
}
```

OUTPUT-

```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
6:Exit
ENTER THE CHOICE
1
Enter the item at front-end
6

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
6:Exit
ENTER THE CHOICE
1
Enter the item at front-end
7

 1:Insert_front
 2:Delete_front
```

```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
6:Exit
ENTER THE CHOICE
1
Enter the item at front-end
8

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
6:Exit
ENTER THE CHOICE
5
8
7
6

 1:Insert_front
```

```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
6:Exit
ENTER THE CHOICE
2
item deleted at front-end is=8

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
6:Exit
ENTER THE CHOICE
2
item deleted at front-end is=7

 1:Insert_front
 2:Delete_front
```

```
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
6:Exit
ENTER THE CHOICE
3
Enter the item at rear-end
4

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
6:Exit
ENTER THE CHOICE
3
Enter the item at rear-end
7

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
```

```
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
6:Exit
ENTER THE CHOICE
3
Enter the item at rear-end
9

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
6:Exit
ENTER THE CHOICE
5
4
7
9

 1:Insert_front
 2:Delete_front
 3:Insert_rear
```

```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
6:Exit
ENTER THE CHOICE
4
item deleted at rear-end is 9
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
6:Exit
ENTER THE CHOICE
4
item deleted at rear-end is 7
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
6:Exit
ENTER THE CHOICE
```

```
ENTER THE CHOICE
4
item deleted at rear-end is 7
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
6:Exit
ENTER THE CHOICE
4
item deleted is 4

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
6:Exit
ENTER THE CHOICE
6


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

LAB-6

WAP to implement Single Linked List with the following operations a]create a link list b]deletion of first element, specified element and last element c]insertion of a node at first position, any position and last element d]display the contents of the list.

CODE-

```
#include<stdio.h>
struct node
{
int data;
struct node *next;
};
struct node *head=NULL;
int length=0;
void insertend(int ele)
{
struct node *newnode,*temp;
newnode=(struct node*)malloc(sizeof(struct node));
newnode->data=ele;
newnode->next=NULL;
if(head==NULL)
{
head=newnode;
length=1;
```

```
}
else
{
temp=(struct node*)malloc(sizeof(struct node));
temp=head;
while(temp->next!=NULL)
{
temp=temp->next;
}
temp->next=newnode;
length++;
}

}
void insertfront(int ele)
{
struct node *temp;
temp=(struct node*)malloc(sizeof(struct node));
temp->data=ele;
temp->next=head;
head=temp;
length++;
```

```
}
void insertrandom(int ele,int pos)
{
if(pos==1)
insertfront(ele);
else if(pos>=length)
insertend(ele);
else
{
struct node *inst;
inst=(struct node*)malloc(sizeof(struct node));
struct node *temp;
temp=(struct node*)malloc(sizeof(struct node));
temp=head;
for(int i=1;i<pos-1;i++)
{
temp=temp->next;
}
inst->data=ele;
inst->next=temp->next;
temp->next=inst;
length++;
```

```
}


}
void deleteele(int ele)
{
struct node *temp,*del;
temp=(struct node*)malloc(sizeof(struct node));
del=(struct node*)malloc(sizeof(struct node));
del=NULL;
if(head->data==ele)
{
del=head;
head=head->next;
del->next=NULL;
}
else
{
temp=head;
while(temp->next!=NULL)
{
if(temp->next->data==ele)
```

```
{


del=temp->next;
temp->next=del->next;
del->next=NULL;
length--;
break;
}
else
{
temp=temp->next;
}
}
}
if(del==NULL)
{
printf("\nElement not found.\n");
}
}
void display()
{
struct node *temp;
```

```
temp=(struct node*)malloc(sizeof(struct node));
temp=head;
if(temp==NULL)
{
printf("\n List is empty \n");
}
else
{
printf("\nThe contents of the list are :\n");
while(temp!=NULL)
{
printf("%d\n",temp->data);
temp=temp->next;
}
}

}
int main()
{
int choice,ele,pos;
char ch;
do
```

```
{
printf("\n1. Inset at end \n2.Insert at front \n3.Insert at random position \n4. Display \n5. Delete
\n6.exit");
printf("\nEnter your choice : ");
scanf("%d",&choice);
switch(choice)
{
case 1: printf("Enter the element to be inserted\n");
scanf("%d",&ele);
insertend(ele);
break;
case 2: printf("Enter the element to be inserted\n");
scanf("%d",&ele);
insertfront(ele);
break;
case 3: printf("Enter the element to be inserted\n");
scanf("%d",&ele);
printf("Enter the position \n");
scanf("%d",&pos);
insertrandom(ele,pos);
break;
```

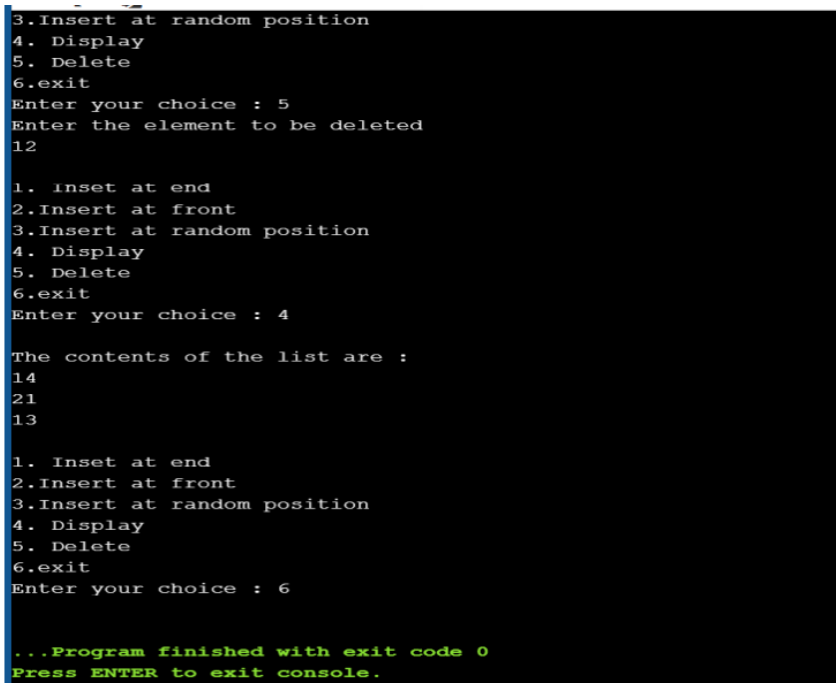
```
case 4: display();
break;
case 5: printf("Enter the element to be deleted\n");
scanf("%d",&ele);
deleteele(ele);
break;
}
}while(choice!=6);
return 0;
}
```

OUTPUT-

```
1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 1
Enter the element to be inserted
12

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 1
Enter the element to be inserted
13
```

```
Enter your choice : 1
Enter the element to be inserted
13

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 2
Enter the element to be inserted
14

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 3
Enter the element to be inserted
21
Enter the position
2

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
```

```
The contents of the list are :
14
21
12
13

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 5
Enter the element to be deleted
11

Element not found.

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 5
Enter the element to be deleted
12

1. Inset at end
2.Insert at front
3.Insert at random position
```



LAB-7

WAP to implement Single Linked List with the following operations a] sort the linked list b]reverse the linked list c]concatenation of 2 linked lists

CODE-

```
#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("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;
}
int main()
{
int item,choice,pos,i,n;
NODE first=NULL,a,b;
```

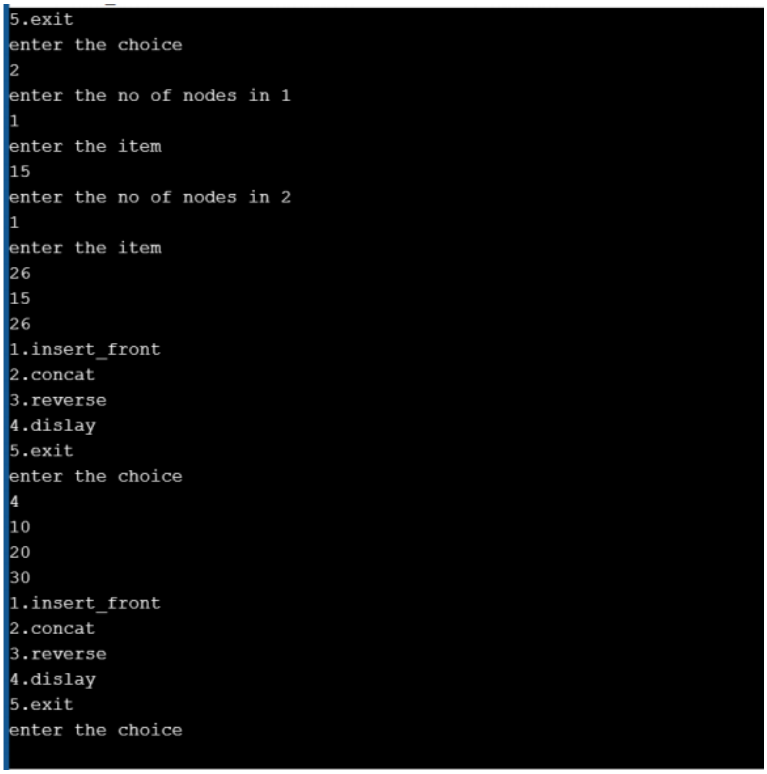
```
for(;;)
{
printf("1.insert_front\n2.concat\n3.reverse\n4.dislay\n5.exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the item\n");
scanf("%d",&item);
first=insert_rear(first,item);
break;
case 2:printf("enter the no of nodes in 1\n");
scanf("%d",&n);
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:display(first);
break;
default:exit(0);
}
}
}
```

OUTPUT-

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

```
enter the choice
3
10
20
30
1.insert_front
2.concat
3.reverse
4.dislay
5.exit
enter the choice
2
enter the no of nodes in 1
1
enter the item
15
enter the no of nodes in 2
1
enter the item
26
15
26
1.insert_front
2.concat
3.reverse
4.dislay
5.exit
enter the choice
4
10
```



LAB-8

WAP to implement stack and queues using linked representation

CODE-

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *next;
```

```
};
typedef struct node *NODE;
NODE get()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{
printf("Memory full\n");
exit(0);
}
return x;
}
void f(NODE x)
{
free(x);
}
NODE insert_front(NODE first, int item)
{
NODE temp;
temp=get();
temp->data=item;
```

```
temp->next=NULL;
if(first==NULL)
{
return temp;
}
temp->next=first;
first=temp;
return first;
}
NODE delete_front(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("List Empty\n");
return first;
}
temp=first;
temp=temp->next;
printf("%d is deleted",first->data);
f(first);
return temp;
```

```
}
NODE insert_rear(NODE first, int item)
{
NODE temp,cur;
temp=get();
temp->data=item;
temp->next=NULL;
if(first==NULL)
{
return temp;
}
cur=first;
while(cur->next!=NULL)
{
cur=cur->next;
cur->next=temp;
}
return first;
}
NODE delete_rear(NODE first)
{
NODE cur,prev;
```

```
if(first==NULL)
{
printf("List Empty\n");
return first;
}
if(first->next==NULL)
{
printf("%d is deleted\n",first->data);
f(first);
return NULL;
}
prev=NULL;
cur=first;
while(cur->next!=NULL)
{
prev=cur;
cur=cur->next;
}
printf("Item deleted=%d\n",cur->data);
f(cur);
prev->next=NULL;
return first;
```

```
}
void display(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("List Empty\n");
}
for(temp=first;temp!=NULL;temp=temp->next)
{
printf("%d\n",temp->data);
}
}
int main()
{
int item,ch;
NODE first=NULL;
do{
printf("1.Insert front\n2.Delete front\n3.Insert rear\n4.Delete
rear\n5.Display\n6.Exit\n");
printf("Enter choice: ");
scanf("%d",&ch);
```

```
switch(ch)
{
case 1:
printf("Enter item at front: ");
scanf("%d",&item);
first=insert_front(first,item);
break;
case 2:
first=delete_front(first);
break;
case 3:
printf("Enter item at rear: ");
scanf("%d",&item);
first=insert_rear(first,item);
break;
case 4:
first=delete_rear(first);
break;
case 5:
display(first);
break;
case 6:
```

```
break;

default:

printf("\nEnter a valid choice\n");

break;

}

}while(ch!=6);

return 0;

}
```

OUTPUT-

LAB-9

WAP to Implement doubly link list with primitive operations a) Create a doubly linked list. b) Insert a new node to the left of the node. c) Delete the node based on a specific value d) Display the contents of the list e) Delete the duplicates [plus other functions]

CODE-

```
#include<stdio.h>
#include<stdlib.h>
#include<process.h>
struct node
{
int info;
struct node *rlink;
struct node *llink;
};
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_rear(NODE head,int item)
{
NODE temp,cur;
temp=getnode();
temp->rlink=NULL;
temp->llink=NULL;
temp->info=item;
cur=head->llink;
temp->llink=cur;
cur->rlink=temp;
head->llink=temp;
temp->rlink=head;
head->info=head->info+1;
return head;
```

```
}
NODE insert_leftpos(int item,NODE head)
{
NODE temp,cur,prev;
if(head->rlink==head)
{
printf("list empty\n");
return head;
}
cur=head->rlink;
while(cur!=head)
{
if(item==cur->info)break;
cur=cur->rlink;
}
if(cur==head)
{
printf("key not found\n");
return head;
}
prev=cur->llink;
printf("enter towards left of %d=",item);
```

```
temp=getnode();
scanf("%d",&temp->info);
prev->rlink=temp;
temp->llink=prev;
cur->llink=temp;
temp->rlink=cur;
return head;
}
NODE insert_righttpos(int item,NODE head)
{
NODE temp,cur,prev;
if(head->rlink==head)
{
printf("list empty\n");
return head;
}
cur=head->rlink;
while(cur!=head)
{
if(item==cur->info)break;
cur=cur->rlink;
}
```

```
if(cur==head)
{
printf("key not found\n");
return head;
}
prev=cur->rlink;
printf("enter towards left of %d=",item);
temp=getnode();
scanf("%d",&temp->info);
prev->llink=temp;
temp->llink=cur;
cur->rlink=temp;
temp->rlink=prev;
return head;
}
NODE delete_all_key(int item,NODE head)
{
NODE prev,cur,next;
int count;
if(head->rlink==head)
{
printf("LE");
```

```
return head;
}
count=0;
cur=head->rlink;
while(cur!=head)
{
if(item!=cur->info)
cur=cur->rlink;
else
{
count++;
prev=cur->llink;
next=cur->rlink;
prev->rlink=next;
next->llink=prev;
freenode(cur);
cur=next;
}
}
if(count==0)
printf("key not found");
else
```

```
printf("key found at %d positions and are deleted\n", count);
return head;
}
void Search_info(int item,NODE head){
NODE cur;
if(head->rlink==head)
{
printf("list empty\n");
}
cur=head->rlink;
while(cur!=head)
{
if(item==cur->info)
{
printf("Search Successfull\n");
break;
}
cur=cur->rlink;
}
if(cur==head)
{
printf("Info not found\n");
```

```
}
}
void display(NODE head)
{
NODE temp;
if(head->rlink==head)
{
printf("list empty\n");
return;
}
for(temp=head->rlink;temp!=head;temp=temp->rlink)
printf("%d\n",temp->info);
}
void main()
{
int item,choice,key;
NODE head;
head=getnode();
head->rlink=head;
head->llink=head;
for(;;)
{
```

```
printf("\n1.insert_rear\n2.insert_key_left\n3.insert_key_right\n4.delete_duplic
ates\n5.Searh_info\n6.d

isplay\n7.exit\n");

printf("enter the choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:printf("enter the item\n");

scanf("%d",&item);

head=insert_rear(head,item);

break;

case 2:printf("enter the key item\n");

scanf("%d",&item);

head=insert_leftpos(item,head);

break;

case 3:printf("enter the key item\n");

scanf("%d",&item);

head=insert_righttpos(item,head);

break;

case 4:printf("enter the key item\n");

scanf("%d",&item);

head=delete_all_key(item,head);
```

```
break;
case 5:printf("enter the key item\n");
scanf("%d",&item);
Search_info(item,head);
break;
case 6:display(head);
break;
default:exit(0);
break;
}
}
}
```

OUTPUT-

```
1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
1
enter the item
10

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
1
enter the item
10

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
```

```
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
1
enter the item
20

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
6
10
10
20

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
6
10
```

```
7.exit
enter the choice
6
10
10
20

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
1
enter the item
15

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
1
enter the item
20

1.insert_rear
```

```
7.exit
enter the choice
6
10
10
20

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
1
enter the item
15

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
1
enter the item
20

1.insert_rear
```

```
7.exit
enter the choice
1
enter the item
20

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
5
enter the key item
15
Search Successfull

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
7


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

LAB-10

Write a program a]To construct a binary search tree b]to traverse the tree using methods i.e. inorder, preorder, postorder c]to display the elements of the tree

CODE-

```
#include<stdio.h>

#include<conio.h>

#include<process.h>

#include<stdlib.h>

struct node

{

int info;

struct node *rlink;

struct node *llink;

};

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(NODE root,int item)
{
NODE temp,cur,prev;
temp=getnode();
temp->rlink=NULL;
temp->llink=NULL;
temp->info=item;
if(root==NULL)
return temp;
prev=NULL;
```

```
cur=root;
while(cur!=NULL)
{
prev=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(item<prev->info)
prev->llink=temp;
else
prev->rlink=temp;
return root;
}
void display(NODE root,int i)
{
int j;
if(root!=NULL)
{
display(root->rlink,i+1);
for(j=0;j<i;j++)
printf(" ");
```

```
printf("%d\n",root->info);
display(root->llink,i+1);
}
}
NODE delete(NODE root,int item)
{
NODE cur,parent,q,suc;
if(root==NULL)
{
printf("empty\n");
return root;
}
parent=NULL;
cur=root;
while(cur!=NULL&&item!=cur->info)
{
parent=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(cur==NULL)
```

```
{
printf("not found\n");
return root;
}
if(cur->llink==NULL)
q=cur->rlink;
else if(cur->rlink==NULL)
q=cur->llink;
else
{
suc=cur->rlink;
while(suc->llink!=NULL)
suc=suc->llink;
suc->llink=cur->llink;
q=cur->rlink;
}
if(parent==NULL)
return q;
if(cur==parent->llink)
parent->llink=q;
```

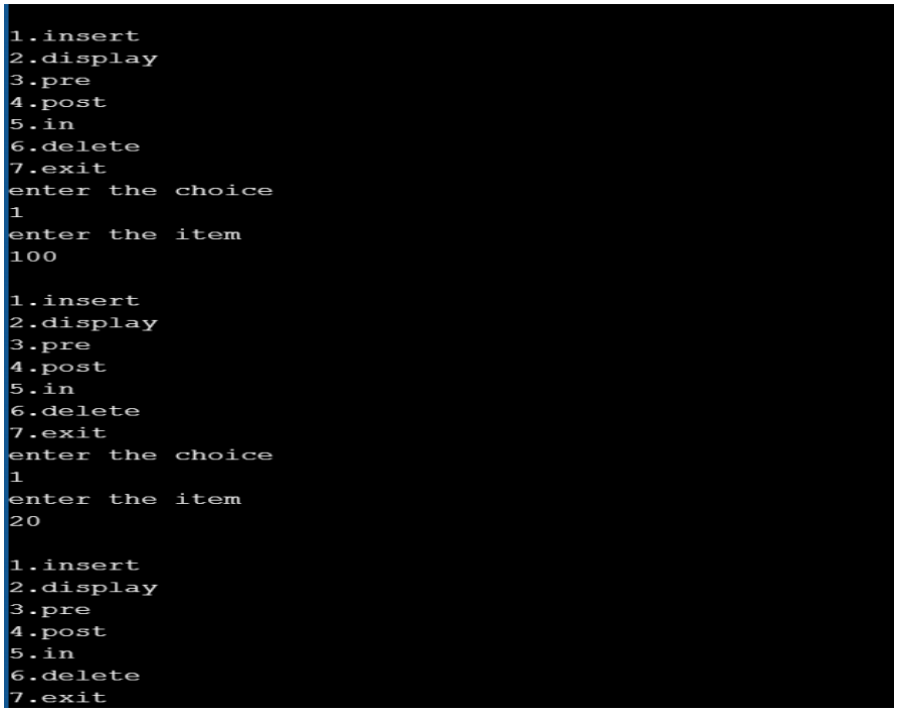
```
else
parent->rlink=q;
freenode(cur);
return root;
}
void preorder(NODE root)
{
if(root!=NULL)
{
printf("%d\n",root->info);
preorder(root->llink);
preorder(root->rlink);
}
}
void postorder(NODE root)
{
if(root!=NULL)
{
postorder(root->llink);
postorder(root->rlink);
```

```
printf("%d\n",root->info);
}
}
void inorder(NODE root)
{
if(root!=NULL)
{
inorder(root->llink);
printf("%d\n",root->info);
inorder(root->rlink);
}
}
void main()
{
int item,choice;
NODE root=NULL;
for(;;)
{
printf("\n1.insert\n2.display\n3.pre\n4.post\n5.in\n6.delete\n7.exit\
n");
```

```
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the item\n");
scanf("%d",&item);
root=insert(root,item);
break;
case 2:display(root,0);
break;
case 3:preorder(root);
break;
case 4:postorder(root);
break;
case 5:inorder(root);
break;
case 6:printf("enter the item\n");
scanf("%d",&item);
root=delete(root,item);
break;
```

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

OUTPUT-

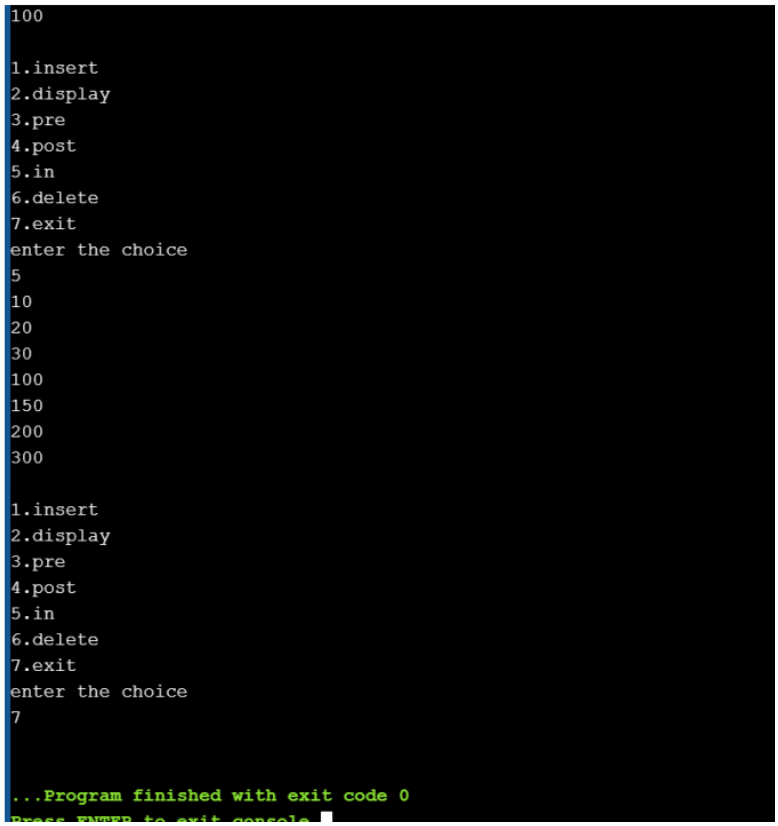


```
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
200

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
10

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
30
```

```
enter the choice
1
enter the item
30

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
150

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
300

1.insert
2.display
3.pre
```

```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
    300
  200
    150
100
    30
  20
    10

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
3
100
20
10
30
200
```

```
20
10
30
200
150
300

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
4
10
30
20
150
300
200
100

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
```


100
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
5
10
20
30
100
150
200
300
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
7
...Program finished with exit code 0