BANGALORE UNIVERSITY

UNIVERSITY VISVESWARAYA COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

B.Tech. PROGRAME IN INFORMATION SCIENCE AND ENGINEERING

Course Code: 18CIPC308

Course Title: DATA STRUCTURES AND APPLICATIONS LABORATORY

NAME :

CLASS :

Reg No :

UNDER THE GUIDENCE OF: **SIGNATURE:**

1. **Dr. LATA BT**
2. **SHRUTHI T**

CONTENTS

Write a Program in C to:

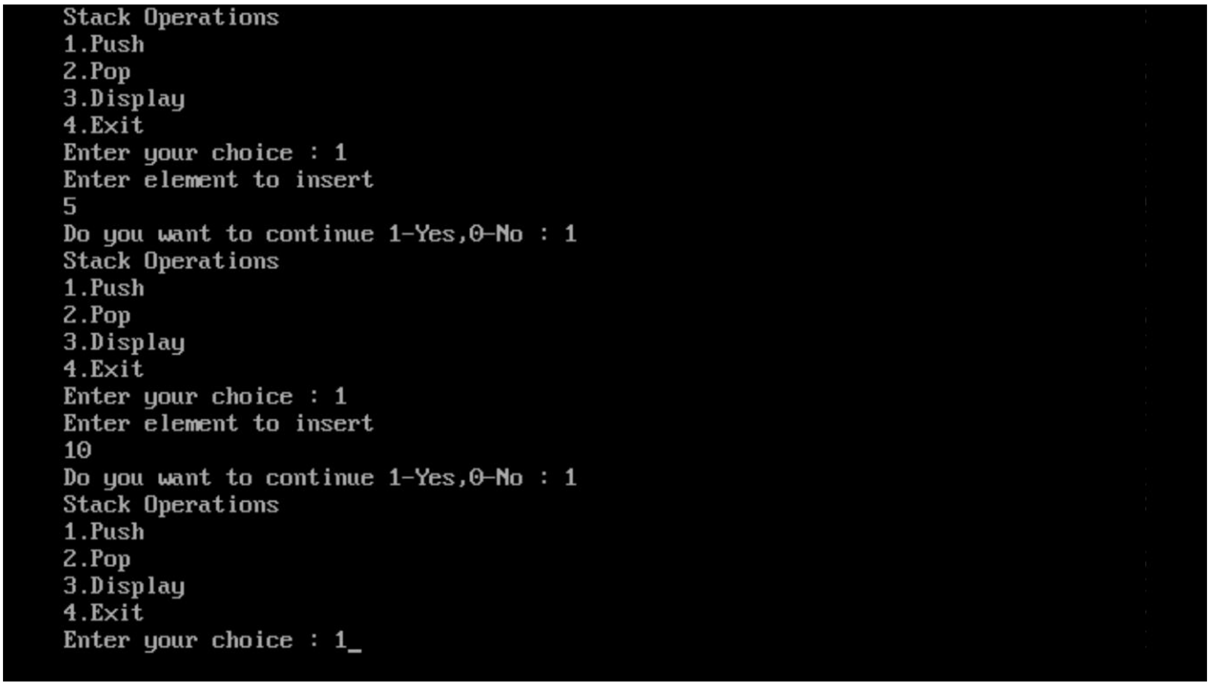
Program Number	Program Name	PAGE NO
1	Implement the following using Arrays:	
	a. Stack.	1 – 4
	b. Queue.	5 – 8
2	Use Recursive program to implement :	
	a. Tower of Hanoi.	9 – 10
	b. Insertion Sort.	11 – 12
3	Convert infix expression to prefix expression.	13 – 16
4	Implement Singly Circular Linked List with header node.	17 – 28
5	Implement Double Ended Queue using Singly Linked List.	29 – 37
6	Perform various operations in Doubly linked list.	38 – 49
7	Create a binary tree and traverse in Inorder, Preorder and Postorder.	50 – 54
8	Evaluate expression tree using binary tree.	55 – 57
9	Perform insert and delete operation in Binary search tree.	58 – 63
10	Create right in threaded binary tree.	64 – 69
11	Implement Hash tables:	
12	Implement Hashing using open addressing.	
13	Write all the members of an array of structures to a file using fwrite(). Read the array from file and display on the screen.	
14	Compare the contents of two files. Write the difference in another file.	

1a. Stack using array.

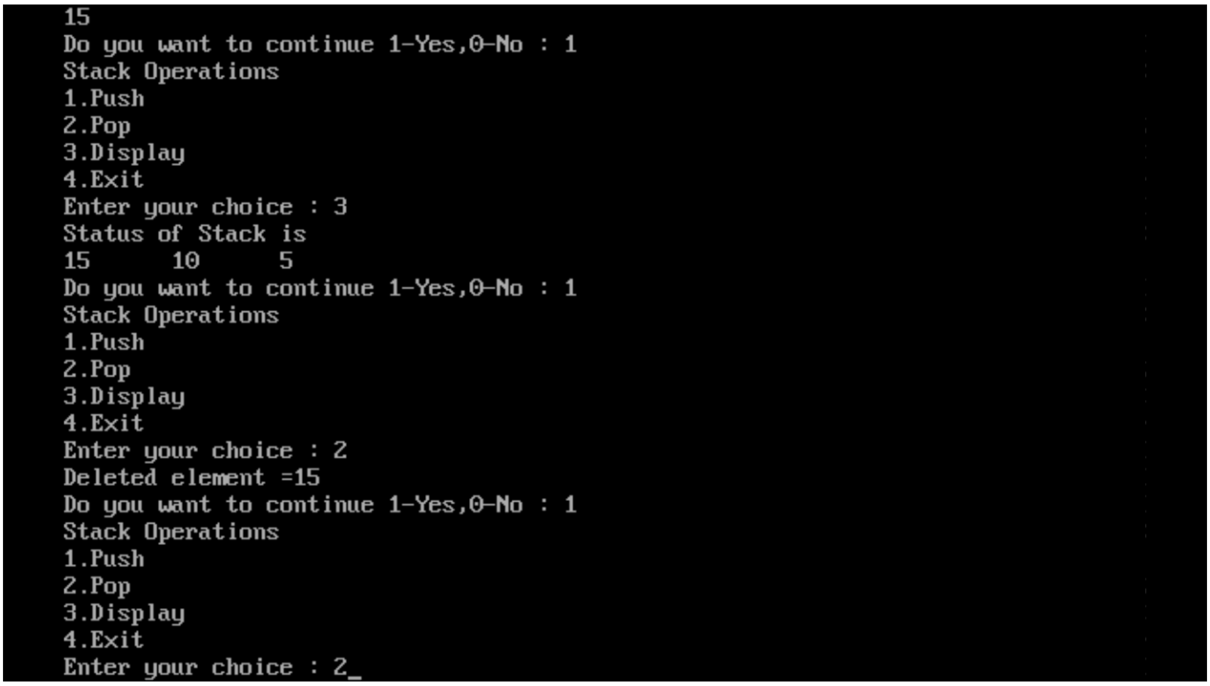
```
#include<stdio.h>
#include<conio.h>
#define SIZE 10
int top=-1,stack[20];
void main()
{
	int ch=1,option;
	clrscr();
	while(ch==1)
	{
		printf("Stack Operations\n");
		printf("1.Push\n2.Pop\n3.Display\n4.Exit\n");
		printf("Enter your choice : ");
		scanf("%d",&option);
		switch(option)
		{
			case 1:push();break;
			case 2:pop();break;
			case 3:display();break;
			case 4:exit(0);break;
			default :printf("Wrong choice\n");
		}
		printf("Do you want to continue 1-Yes,0-No : ");
		scanf("%d",&ch);
	}
}
int push()
{
        int num;
        if(top==(SIZE-1))
        {
                printf("Stack is full\n");
                return;
        }
        else
        {
                printf("Enter element to insert\n");
                scanf("%d",&num);
                stack[++top]=num;
                return;
        }
}
int pop()
{
        if(top==-1)
        {
                printf("Stack is empty\n");
                return;
        }
        else
        {
                printf("Deleted element =%d\n",stack[top]);
                top--;
        }
        return;
}
int display()
```

```
{
        int i;
        if(top==-1)
        {
                printf("Stack is empty\n");
                return;
        }
        else
        {
                printf("Status of Stack is\n");
                for(i=top;i>=0;i--)
                        printf("%d\t",stack[i]);printf("\n");
                return;
        }
}
```

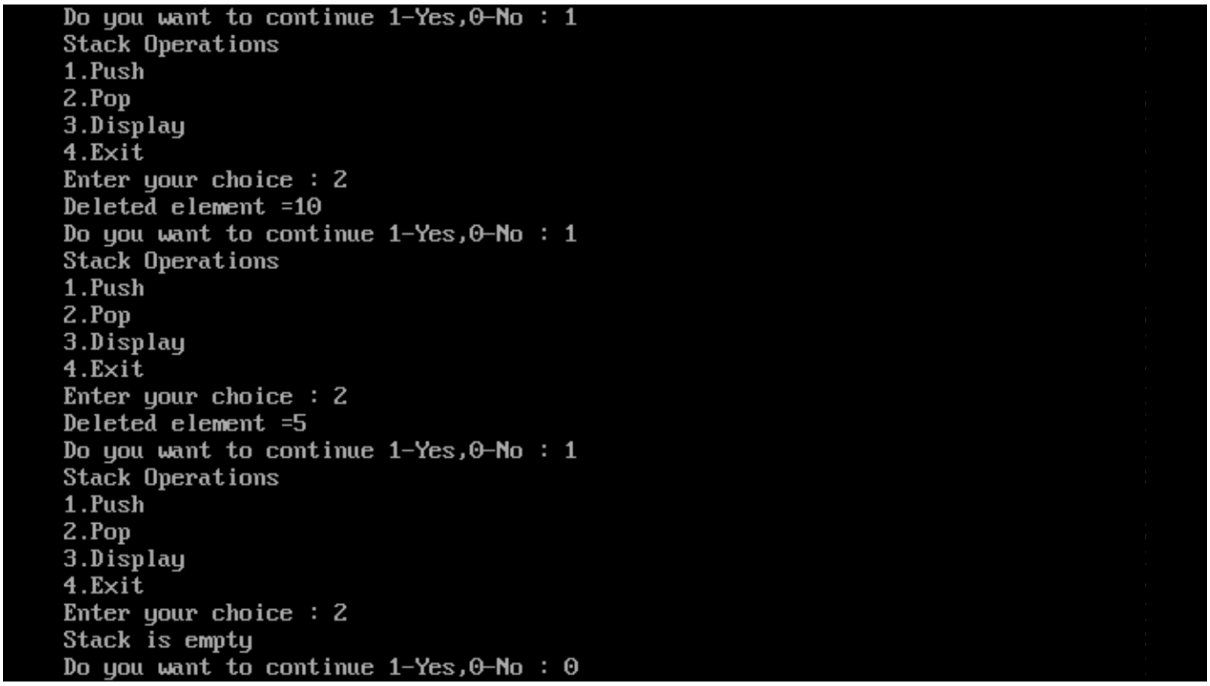
OUTPUT:



```
Stack Operations
1.Push
2.Pop
3.Display
4.Exit
Enter your choice : 1
Enter element to insert
5
Do you want to continue 1-Yes,0-No : 1
Stack Operations
1.Push
2.Pop
3.Display
4.Exit
Enter your choice : 1
Enter element to insert
10
Do you want to continue 1-Yes,0-No : 1
Stack Operations
1.Push
2.Pop
3.Display
4.Exit
Enter your choice : 1_
```



```
15
Do you want to continue 1-Yes,0-No : 1
Stack Operations
1.Push
2.Pop
3.Display
4.Exit
Enter your choice : 3
Status of Stack is
15      10      5
Do you want to continue 1-Yes,0-No : 1
Stack Operations
1.Push
2.Pop
3.Display
4.Exit
Enter your choice : 2
Deleted element =15
Do you want to continue 1-Yes,0-No : 1
Stack Operations
1.Push
2.Pop
3.Display
4.Exit
Enter your choice : 2_
```



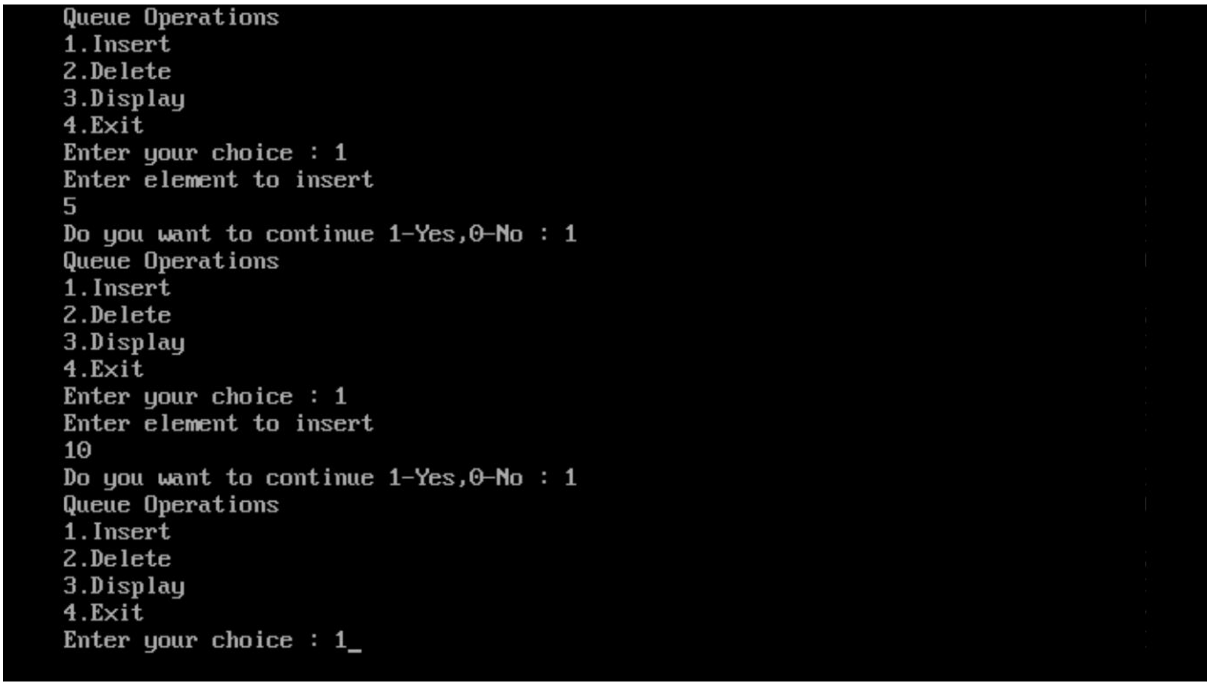
```
Do you want to continue 1-Yes,0-No : 1
Stack Operations
1.Push
2.Pop
3.Display
4.Exit
Enter your choice : 2
Deleted element =10
Do you want to continue 1-Yes,0-No : 1
Stack Operations
1.Push
2.Pop
3.Display
4.Exit
Enter your choice : 2
Deleted element =5
Do you want to continue 1-Yes,0-No : 1
Stack Operations
1.Push
2.Pop
3.Display
4.Exit
Enter your choice : 2
Stack is empty
Do you want to continue 1-Yes,0-No : 0
```

1b. Queue using array.

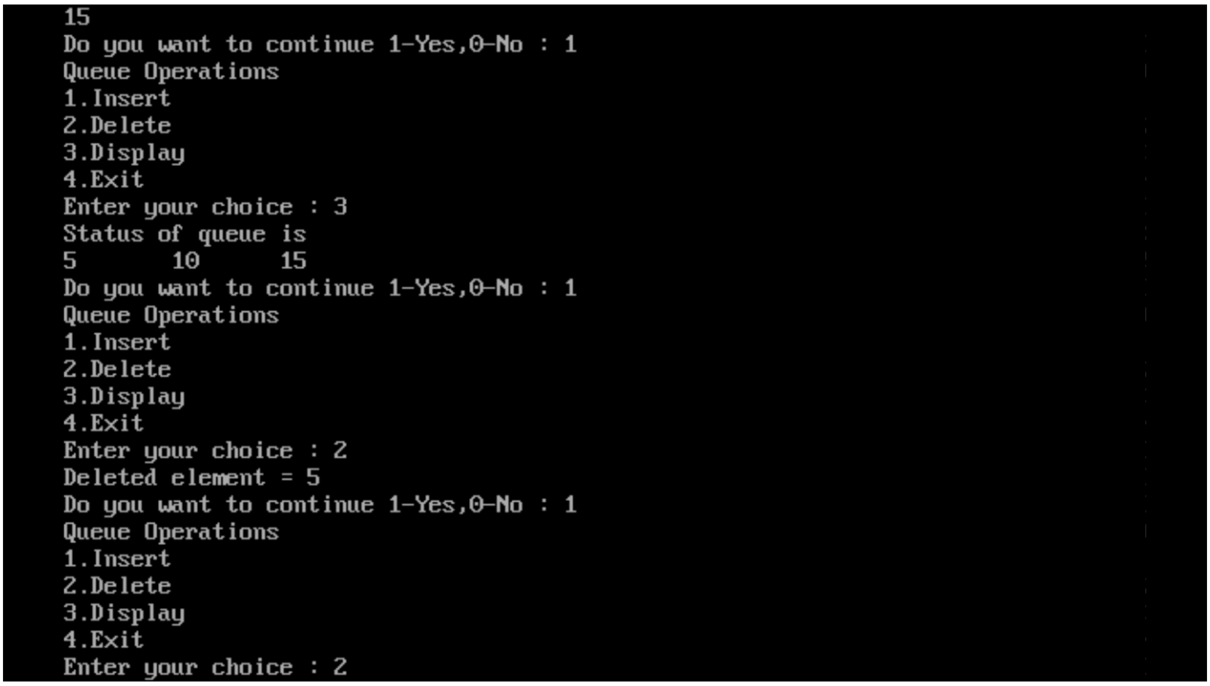
```
#include<stdio.h>
#include<conio.h>
#define SIZE 20
int rear=-1,front=-1,queue[10];
void main()
{
        int ch=1,option;
        clrscr();
        while(ch==1)
        {
                printf("Queue Operations\n");
                printf("1.Insert\n2.Delete\n3.Display\n4.Exit\n");
                printf("Enter your choice \n");
                scanf("%d",&option);
                switch(option)
                {
                        case 1:qinsert();break;
                        case 2:qdelete();break;
                        case 3:qdisplay();break;
                        case 4:exit(0);
                        default :printf("Wrong choice\n");
                }
                printf("Do you want to continue 1-Yes,0-No\n");
                scanf("%d",&ch);
        }
}
int qinsert()
{
        int num;
        if(rear==(SIZE-1))
        {
                printf("Queue is full\n");
                return;
        }
        printf("Enter element to insert\n");
        scanf("%d",&num);
        queue[++rear]=num;
        if(front==-1)
                front++;
        return;
}
int qdelete()
{
        if(front==-1)
        {
                printf("Queue is Empty\n");
                return;
        }
        if(front==rear)
        {
                printf("Deleted element = %d\n",queue[front]);
                front=-1;rear=-1;
                return;
        }
        printf("Deleted element = %d\n",queue[front]);
        front++;
        return;
```

```
}
int qdisplay()
{
	int i;
	if(front==-1)
	{
		printf("Queue is Empty\n");
		return;
	}
	printf("Status of queue is\n");
	for(i=front;i<=rear;i++)
		printf("%d\t",queue[i]);
	printf("\n");
	return;
}
```

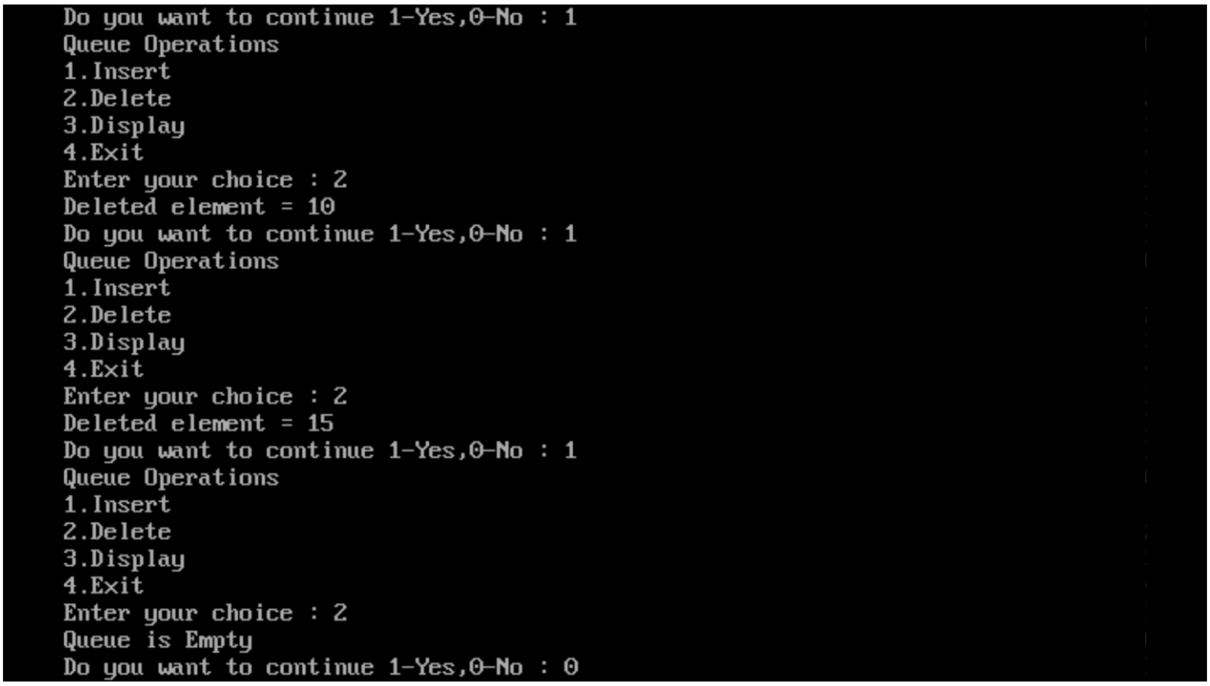
OUTPUT:



```
Queue Operations
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice : 1
Enter element to insert
5
Do you want to continue 1-Yes,0-No : 1
Queue Operations
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice : 1
Enter element to insert
10
Do you want to continue 1-Yes,0-No : 1
Queue Operations
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice : 1_
```



```
15
Do you want to continue 1-Yes,0-No : 1
Queue Operations
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice : 3
Status of queue is
5       10      15
Do you want to continue 1-Yes,0-No : 1
Queue Operations
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice : 2
Deleted element = 5
Do you want to continue 1-Yes,0-No : 1
Queue Operations
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice : 2
```

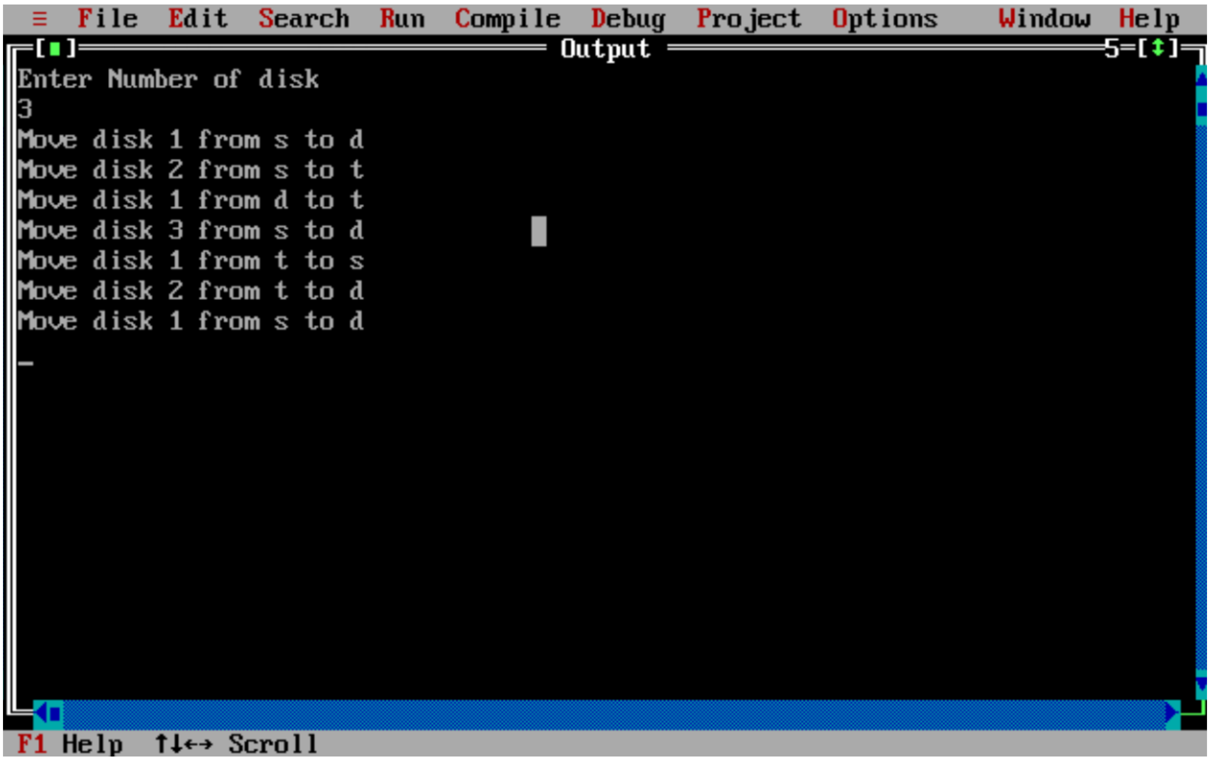


```
Do you want to continue 1-Yes,0-No : 1
Queue Operations
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice : 2
Deleted element = 10
Do you want to continue 1-Yes,0-No : 1
Queue Operations
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice : 2
Deleted element = 15
Do you want to continue 1-Yes,0-No : 1
Queue Operations
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice : 2
Queue is Empty
Do you want to continue 1-Yes,0-No : 0
```

2a. Tower of Hanoi.

```
#include<stdio.h>

#include<conio.h>

void main()

{

        int n;

        clrscr();

        printf("Enter Number of disk\n");

        scanf("%d",&n);

        tower(n,'s','d','t');

        getch();

}

tower(int n,char source,char dest,char temp)

{

        if(n>0)

        {

                tower((n-1),source,temp,dest);

                printf("Move disk %d from %c to %c\n",n,source,dest);

                tower(n-1,temp,dest,source);

        }

        return;

}
```

OUTPUT:

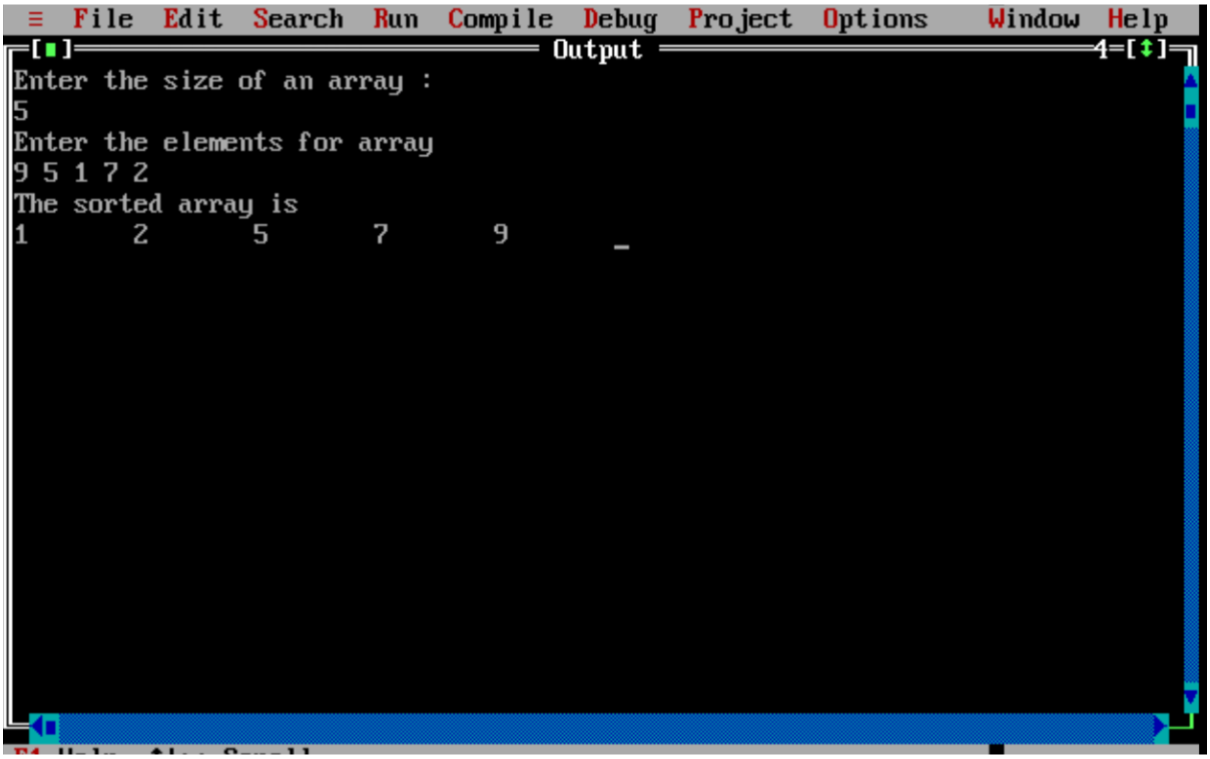

≡ File Edit Search Run Compile Debug Project Options Window Help
[■] Output 5=[↕]
F1 Help ↑↓↔ Scroll

```
Enter Number of disk
3
Move disk 1 from s to d
Move disk 2 from s to t
Move disk 1 from d to t
Move disk 3 from s to d
Move disk 1 from t to s
Move disk 2 from t to d
Move disk 1 from s to d
_
```

2b. Insertion sort.

```
#include<stdio.h>
#include<conio.h>
int a[10],i,j,n,temp;
void isr(int a[],int n)
{
        if(n<=1)
                return;
        isr(a,n-1);
        temp=a[n-1];
        j=n-2;
        while(j>=0&&a[j]>temp)
        {
                a[j+1]=a[j];
                j--;
        }
        a[j+1]=temp;
}

void main()
{
        clrscr();
        printf("Enter the size of an array : \n");
        scanf("%d",&n);
        printf("Enter the elements\n");
        for(i=0;i<n;i++)
                scanf("%d",&a[i]);
        isr(a,n);
        printf("Sorted array is : \n");
        for(i=0;i<n;i++)
                printf("%d\t",a[i]);
        getch();
}
```

OUTPUT:


≡ File Edit Search Run Compile Debug Project Options Window Help
[■]
Output
4=[↕]

```
Enter the size of an array :
5
Enter the elements for array
9 5 1 7 2
The sorted array is
1       2       5       7       9       _
```

3. Infix expression to prefix expression.

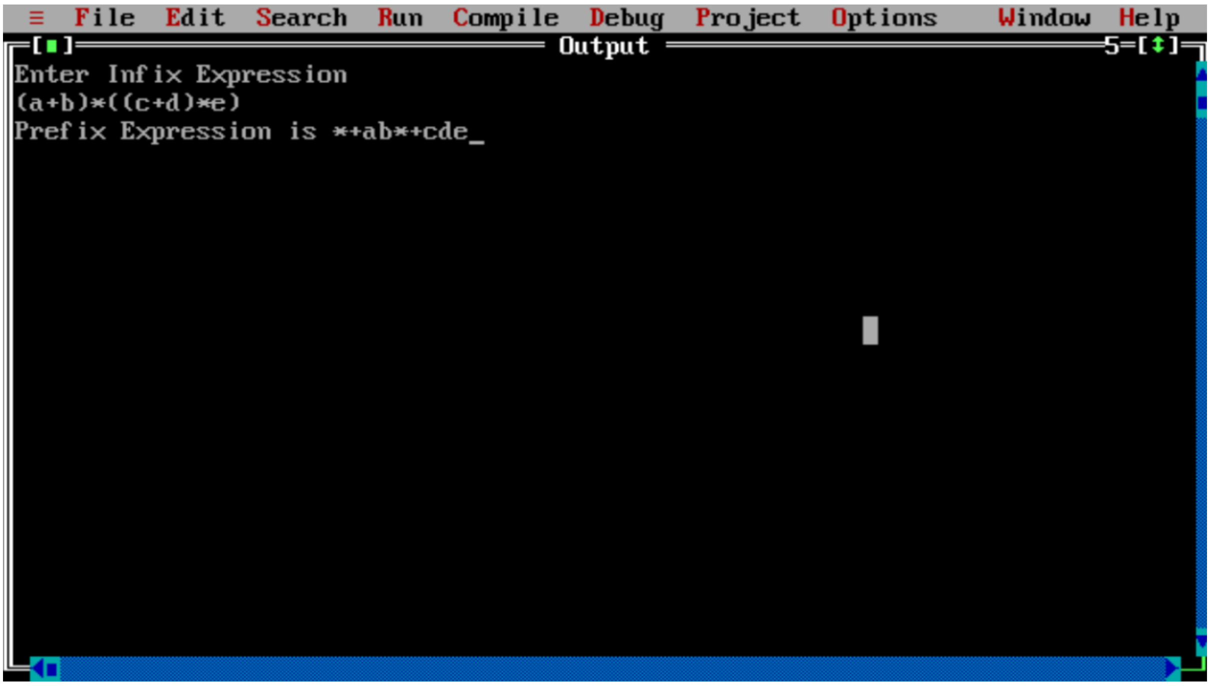
```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void push();
void in_prefix();
char prefix[20],infix[20],stack[20],symbol,temp;
int top=-1,length,i=0,j=0,k=0;
void main()
{
        clrscr();
        printf("Enter Infix Expression\n");
        gets(infix);
        in_prefix(infix,prefix);
        printf("Prefix Expression is %s",prefix);
        getch();
}
void in_prefix(char infix[],char prefix[])
{
        push('#');
        length=strlen(infix);
        j=length;
        while(infix[i]!='\0')
        {
                if(infix[i]=='('||infix[i]==')')
                j--;
                i++;
        }
        while(length>=k)
```

```
{
    symbol=infix[length];
    switch(symbol)
    {
        case')':push(symbol);break;
        case'(':temp=pop();
        while(temp!=')')
        {
            prefix[j--]=temp;
            temp=pop();
        }
        break;
        case'^':
        case'/':
        case'*':
        case'+':
        case'-':while(ISP(stack[top])>=ICP(symbol))
        {
            temp=pop();
            prefix[j--]=temp;
        }
        push(symbol);break;
        default :prefix[j--]=symbol;break;
    }
    length--;
}
while(top>0)
{
    temp=pop();
```

```
                prefix[j--]=temp;
        }
}
void push(char symbol)
{
        stack[++top]=symbol;
        return;
}
pop()
{
        char symbol;
        symbol=stack[top--];
        return(symbol);
}
ISP(char symbol)
{
        int p;
        switch(symbol)
        {
                case'^':p=6;break;
                case'*':
                case'/':p=3;break;
                case'+':
                case'-':p=1;break;
                case')':p=0;break;
                case'#':p=-1;break;
        }
        return(p);
}
```

```
ICP(char symbol)
{
        int q;
        switch(symbol)
        {
                case'^':q=5;break;
                case'*':
                case'/':q=4;break;
                case'+':
                case'-':q=2;break;
                case')':q=0;break;
                case'(':q=9;break;
        }
        return q;
}
```

OUTPUT:


≡ File Edit Search Run Compile Debug Project Options Window Help
Output
5

```
Enter Infix Expression
(a+b)*((c+d)*e)
Prefix Expression is *+ab*+cde_
```

4. Singly Circular linked list with header node.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
	int data;
	struct node *rlink;
};
typedef struct node *NODE;
int i,choice,ch1=1,choice1,ch2=1,choice2,ch3=1;
NODE insert_front(NODE);
NODE insert_end(NODE);
NODE insert_pos(NODE);
NODE delete_front(NODE);
NODE delete_end(NODE);
NODE delete_pos(NODE);
void display(NODE);
void main()
{
	NODE head=0;
	head=(NODE)malloc(sizeof(struct node));
	head->rlink=head;
	clrscr();
	while(ch1==1)
	{
		printf("Circular Singly Linked list implementation\n");
		printf("1:Insert\n2:Delete\n3:Display\n4:Exit\n");
		printf("Enter your choice : ");
```

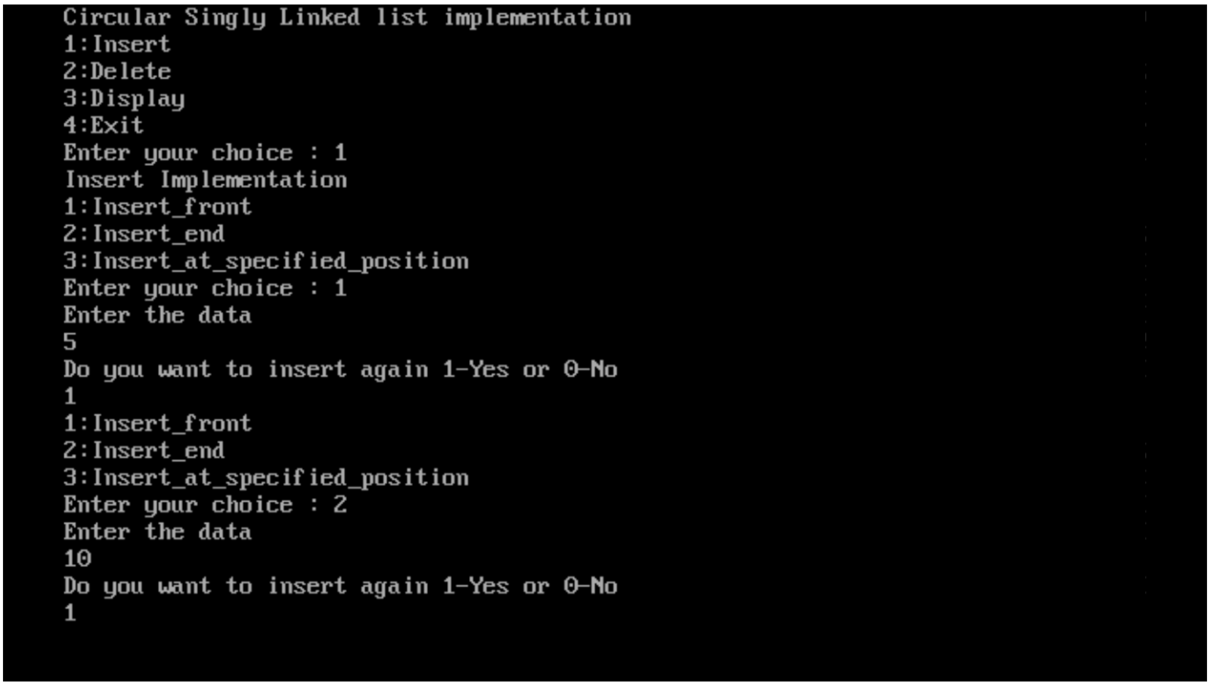
```
        scanf("%d",&choice);
        switch(choice)
        {
                case 1: printf("Insert Implementation\n");
                        ch2=1;
                        while(ch2==1)
                        {
printf("1:Insert_front\n2:Insert_end\n3:Insert_at_specified_position\n");
                                printf("Enter your choice : ");
                                scanf("%d",&choice1);
                                switch(choice1)
                                {
                                        case 1: head=insert_front(head);
                                                break;
                                        case 2: head=insert_end(head);
                                                break;
                                        case 3: head=insert_pos(head);
                                                break;
                                        default : printf("Wrong choice\n");
                                                break;
                                }
                                printf("Do you want to insert again 1-Yes or 0-No\n");
                                scanf("%d",&ch2);
                        }
                        break;
                case 2: printf("Delete Implementation\n");
                        ch3=1;
                        while(ch3==1)
                        {
        }

        printf("1:Delete_front\n2:Delete_end\n3:Delete_at_specified_position\n");
                                printf("Enter your choice : ");
                                scanf("%d",&choice2);
                                switch(choice2)
                                {
                                        case 1: head=delete_front(head);
                                                break;
                                        case 2: head=delete_end(head);
                                                break;
                                        case 3: head=delete_pos(head);
                                                break;
                                        default :printf("Wrong choice\n");
                                                break;
                                }
                                printf("Do you want to delete again 1-Yes or 0-No\n");
                                scanf("%d",&ch3);
                        }
                        break;
                case 3: display(head);
                        break;
                case 4: exit(0);
                default : printf("Wrong choice\n");
                        break;
            }
            printf("Do you want to continue 1-Yes or 0-No\n");
            scanf("%d",&ch1);
        }
        return 0;
}
```

```
NODE insert_front(NODE head)
{
        NODE newnode,first;
        newnode=(NODE)malloc(sizeof(struct node));
        newnode->rlink=0;
        printf("Enter the data\n");
        scanf("%d",&newnode->data);
        if(head->rlink==head)
        {
                head->rlink=newnode;
                newnode->rlink=head;
                return head;
        }
        first=head->rlink;
        head->rlink=newnode;
        newnode->rlink=first;
        return(head);
}

NODE insert_end(NODE head)
{
        NODE newnode,next;
        newnode=(NODE)malloc(sizeof(struct node));
        newnode->rlink=head;
        printf("Enter the data\n");
        scanf("%d",&newnode->data);
        if(head->rlink==head)
        {
                head->rlink=newnode;
        newnode->rlink=head;
        return(head);
    }
    next=head->rlink;
    while(next->rlink!=head)
        next=next->rlink;
    next->rlink=newnode;
    newnode->rlink=head;
     return(head);
}

NODE delete_front(NODE head)
{
    NODE temp,first;
    if(head->rlink==head)
    {
        printf("No nodes in the list\n");
        return(first);
    }
    temp=head->rlink;
    if(temp->rlink==0)
    {
        printf("Deleted data = %d\n",temp->data);
        head->rlink=head;
        free(temp);
        return(head);
    }
    first=temp->rlink;
    printf("Delete data = %d\n",temp->data);
```

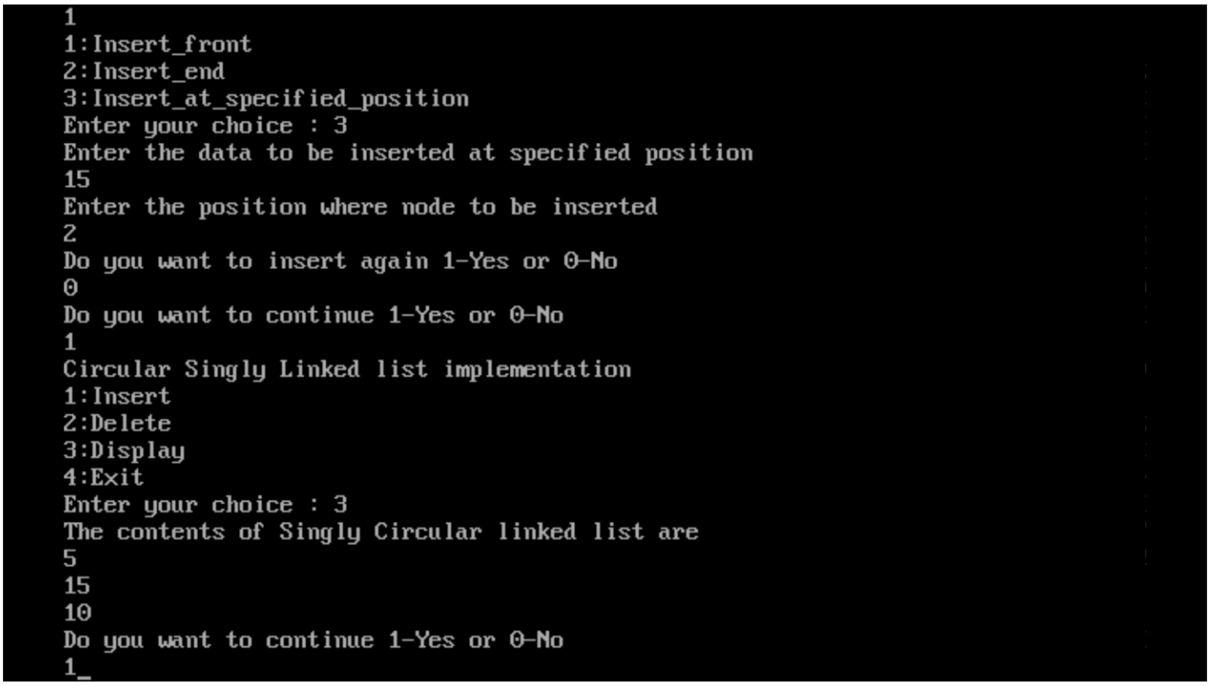
```
        head->rlink=first;
        free(temp);
        return(head);
}

NODE delete_end(NODE head)
{
        NODE temp,prev;
        if(head->rlink==head)
        {
                printf("No nodes in the list\n");
                return(head);
        }
        temp=head->rlink;
        if(temp->rlink==head)
        {
                printf("Deleted data = %d\n",temp->data);
                head->rlink=head;
                free(temp);
                return(head);
        }
        while(temp->rlink!=head)
        {
                prev=temp;
                temp=temp->rlink;
        }
        printf("Deleted data = %d\n",temp->data);
        prev->rlink=head;
        free(temp);
        return(head);
}

void display(NODE head)
{
        NODE next;
        if(head->rlink==head)
        {
                printf("No nodes in the list\n");
        }
        else
        {
                next=head->rlink;
                printf("The contents of Singly Circular linked list are\n");
                while(next!=head)
                {
                        printf("%d  \t",next->data);
                        next=next->rlink;
                        printf("\n");
                }
        }
}

NODE insert_pos(NODE head)/*at specified position*/
{
        NODE newnode,prev,next;
        int i,pos;
        newnode=(NODE)malloc(sizeof(struct node));
        newnode->rlink=newnode;
        printf("Enter the data to be inserted at specified position\n");
        scanf("%d",&newnode->data);
        printf("Enter the position where node to be inserted\n");
        scanf("%d",&pos);
        next=head->rlink;
        if(pos==1)
        {
                head->rlink=newnode;
                newnode->rlink=next;
                return(head);
        }
        for(i=1;i<pos;i++)
        {
                if(next==head)
                {
                        printf("Invalid position\n");
                        return(head);
                }
                prev=next;
                next=next->rlink;
        }/*end of for loop*/
        prev->rlink=newnode;
        newnode->rlink=next;
        return(head);
}

NODE delete_pos(NODE head)
{
        NODE prev,next;
```

```
int i,pos;
if(head->rlink==head)
{
        printf("No nodes in the list\n");
        return(head);
}
printf("Enter the position where the node to be deleted\n");
scanf("%d",&pos);
next=head->rlink;
if(pos==1 && next->rlink==head)
{
        printf("Node to be deleted = %d\n",next->data);
        head->rlink=head;
        free(next);
        return(head);
}
if(pos==1)
{
        printf("Node to be deleted = %d\n",next->data);
        head->rlink=next->rlink;
        free(next);
        return(head);
}
for(i=0;i<pos;i++)
{
        if(next==head)
        {
                printf("Invalid position\n");
                return(head);
                }
                prev=next;
                next=next->rlink;
        }/*end of for loop*/
        printf("Node to be deleted = %d\n",next->data);
        prev->rlink=next->rlink;
        free(next);
        return(head);
}
```

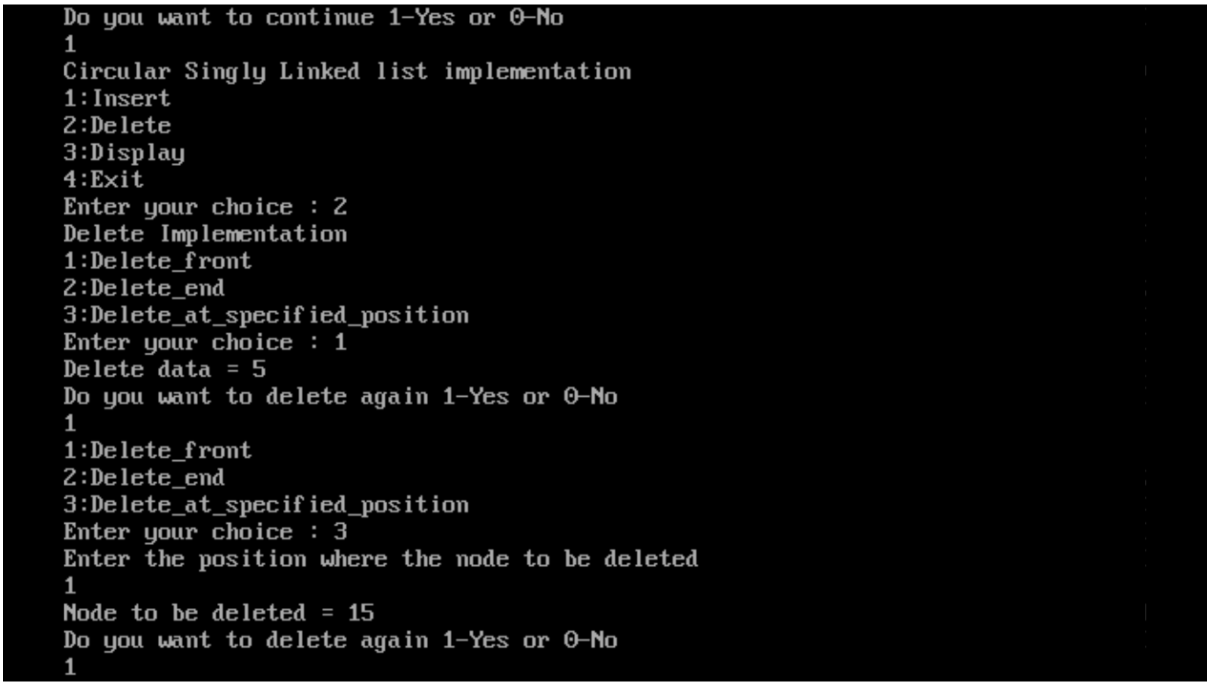
OUTPUT:



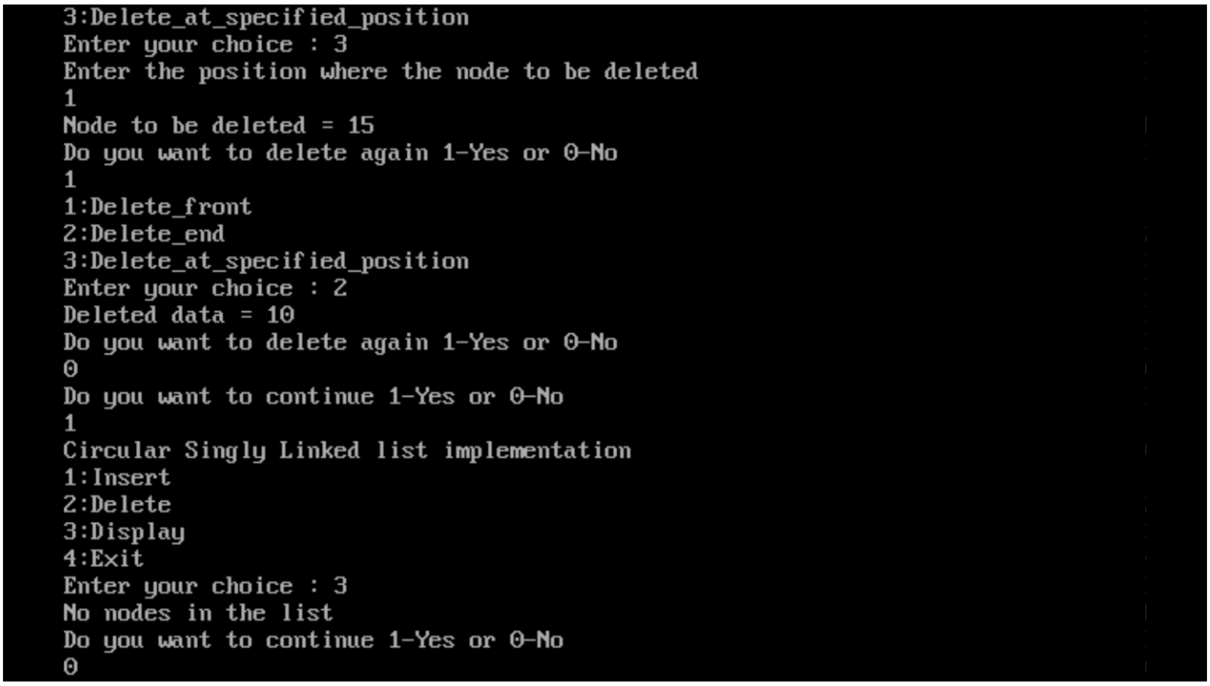
```
Circular Singly Linked list implementation
1:Insert
2:Delete
3:Display
4:Exit
Enter your choice : 1
Insert Implementation
1:Insert_front
2:Insert_end
3:Insert_at_specified_position
Enter your choice : 1
Enter the data
5
Do you want to insert again 1-Yes or 0-No
1
1:Insert_front
2:Insert_end
3:Insert_at_specified_position
Enter your choice : 2
Enter the data
10
Do you want to insert again 1-Yes or 0-No
1
```



```
1
1:Insert_front
2:Insert_end
3:Insert_at_specified_position
Enter your choice : 3
Enter the data to be inserted at specified position
15
Enter the position where node to be inserted
2
Do you want to insert again 1-Yes or 0-No
0
Do you want to continue 1-Yes or 0-No
1
Circular Singly Linked list implementation
1:Insert
2:Delete
3:Display
4:Exit
Enter your choice : 3
The contents of Singly Circular linked list are
5
15
10
Do you want to continue 1-Yes or 0-No
1_
```



```
Do you want to continue 1-Yes or 0-No
1
Circular Singly Linked list implementation
1:Insert
2:Delete
3:Display
4:Exit
Enter your choice : 2
Delete Implementation
1:Delete_front
2:Delete_end
3:Delete_at_specified_position
Enter your choice : 1
Delete data = 5
Do you want to delete again 1-Yes or 0-No
1
1:Delete_front
2:Delete_end
3:Delete_at_specified_position
Enter your choice : 3
Enter the position where the node to be deleted
1
Node to be deleted = 15
Do you want to delete again 1-Yes or 0-No
1
```



```
3:Delete_at_specified_position
Enter your choice : 3
Enter the position where the node to be deleted
1
Node to be deleted = 15
Do you want to delete again 1-Yes or 0-No
1
1:Delete_front
2:Delete_end
3:Delete_at_specified_position
Enter your choice : 2
Deleted data = 10
Do you want to delete again 1-Yes or 0-No
0
Do you want to continue 1-Yes or 0-No
1
Circular Singly Linked list implementation
1:Insert
2:Delete
3:Display
4:Exit
Enter your choice : 3
No nodes in the list
Do you want to continue 1-Yes or 0-No
0
```

5. Double ended queue using singly linked list.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
        int data;
        struct node *link;
};
typedef struct node *NODE;
NODE insert_front(NODE);
NODE insert_rear(NODE);
NODE delete_front(NODE);
NODE delete_rear(NODE);
void display(NODE);
int main()
{
        int ch=1,choice,choice1,p=1,q=1;
        NODE first=0;
        clrscr();
        while(ch==1)
        {
                printf("Double ended queue operation\n");
                printf("1.Input restricted Dequeue\n2.Output restrited
Dequeue\n3:Display\n");
                printf("Enter your choice : ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: p=1;
        while(p==1)
        {
                printf("Input restricted dequeue\n");
                printf("1:Insert from rear\n2:Delete from rear\n3:Delete from front\n");
                printf("Enter your choice : ");
                scanf("%d",&choice1);
                switch(choice1)
                {
                        case 1: first=insert_rear(first);
                                break;
                        case 2: first=delete_rear(first);
                                 break;
                        case 3: first=delete_front(first);
                                 break;
                        default: printf("Invaild choice\n");
                                break;
                }
                printf("Press 1 to continue\n");
                scanf("%d",&p);
        }
        break;
case 2: q=1;
        while(q==1)
        {
                printf("Output restricted Dequeue\n");
                printf("1:Insert from rear\n2:Insert from front\n3:Delete from front\n");
                printf("Enter your choice : ");
                scanf("%d",&choice1);
```

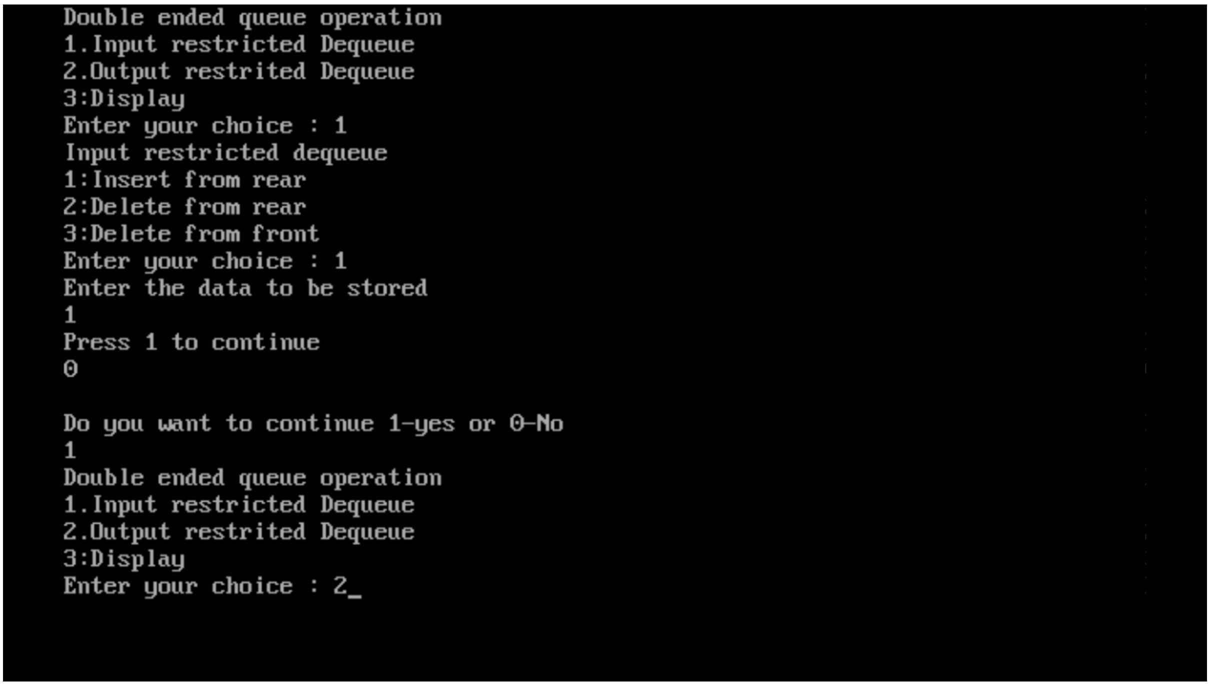
```
                                        switch(choice1)
                                        {
                                                case 1: first=insert_rear(first);
                                                        break;
                                                case 2: first=insert_front(first);
                                                        break;
                                                case 3: first=delete_front(first);
                                                        break;
                                                default: printf("Invalid choice\n");
                                                        break;
                                        }
                                        printf("Press 1 to continue\n");
                                        scanf("%d",&q);
                                }
                                break;
                        case 3:display(first);
                                break;
                        default :printf("Invalid choice\n");
                                break;
                }
                printf("\nDo you want to continue 1-yes or 0-No\n");
                scanf("%d",&ch);
        }
        return;
}

NODE insert_front(NODE first)
{
        NODE newnode;
```

```
	newnode=(NODE)malloc(sizeof(struct node));
	newnode->link=0;
	printf("Enter the data to be stored\n");
	scanf("%d",&newnode->data);
	if(first==0)
	{
		first=newnode;return(first);
	}
	else
	{
		newnode->link=first;
		first=newnode;return(first);
	}
}

NODE delete_front(NODE first)
{
	NODE temp;
	if(first==0)
	{
		printf("No nodes present in list\n");
		return(first);
	}
	else
	{
		printf("Node to be deleted = %d\n",first->data);
		temp=first;
		if(first->link==0)
		{
```

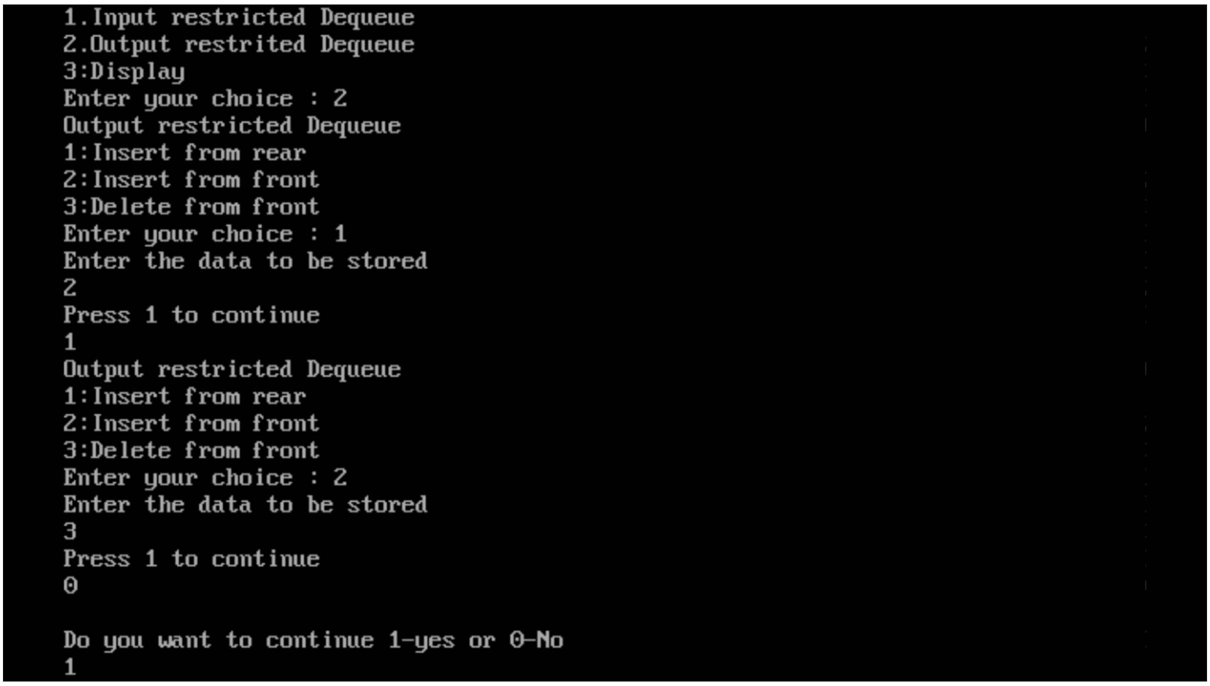
```
                        free(first);
                        first=0;return(first);
                }
                else
                {
                        first=first->link;free(temp);
                        return(first);
                }
        }
}

NODE insert_rear(NODE first)
{
        NODE newnode,temp;
        newnode=(NODE)malloc(sizeof(struct node));
        newnode->link=0;
        printf("Enter the data to be stored\n");
        scanf("%d",&newnode->data);
        if(first==0)
        {
                first=newnode;return(first);
        }
        else
        {
                temp=first;
                while(temp->link!=0)
                        temp=temp->link;
                temp->link=newnode;
                return(first);
        }
}

NODE delete_rear(NODE first)
{
        NODE temp,temp1;
        if(first==0)
        {
                printf("No nodes present in list\n");
                return(first);
        }
        else
        {
                temp=first;
                if(temp->link==0)
                {
                        printf("Node to be deleted = %d\n",temp->data);
                        free(temp);first=0;return(first);
                }
                while(temp->link!=0)
                {
                        temp1=temp;
                        temp=temp->link;
                }
                printf("Node to deleted = %d\n",temp->data);
                free(temp);
                temp1->link=0;
                return(first);
        }
```

```
}

void display(NODE first)
{
        NODE temp;
        if(first==0)
                printf("No nodes are present in list\n");
        else
        {
                printf("The contents of the list are\n");
                for(temp=first;temp!=0;temp=temp->link)
                        printf("%d\t",temp->data);
        }
        printf("\n");
}
```

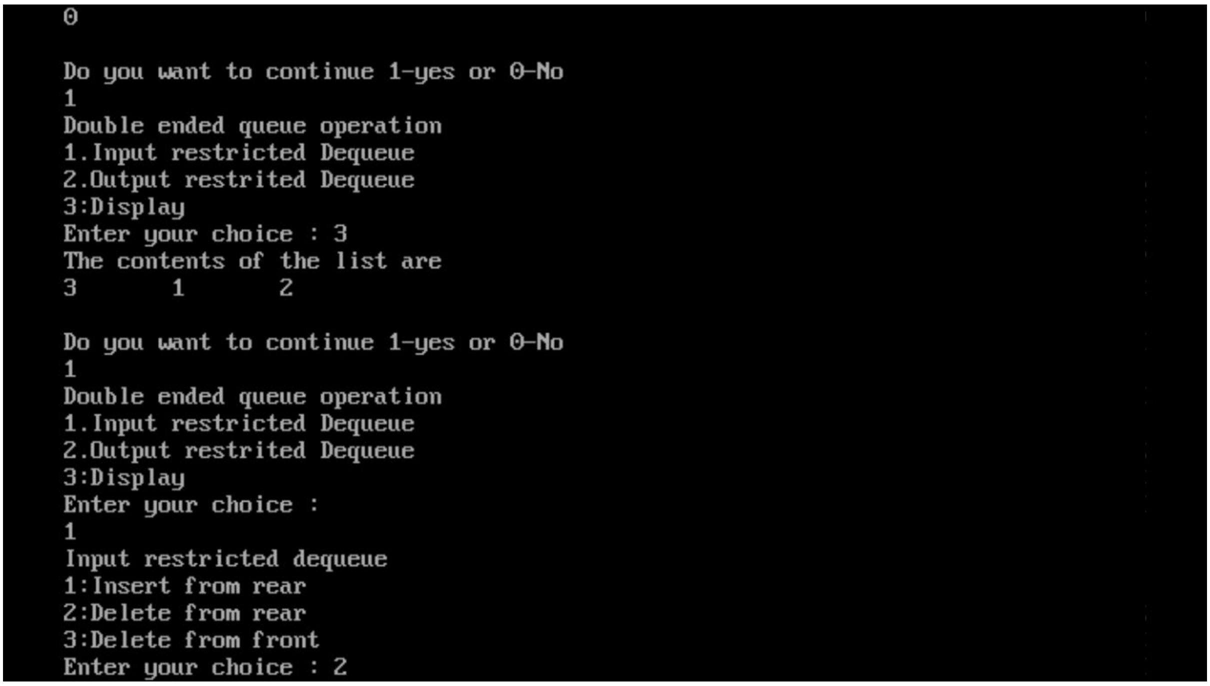
OUTPUT:



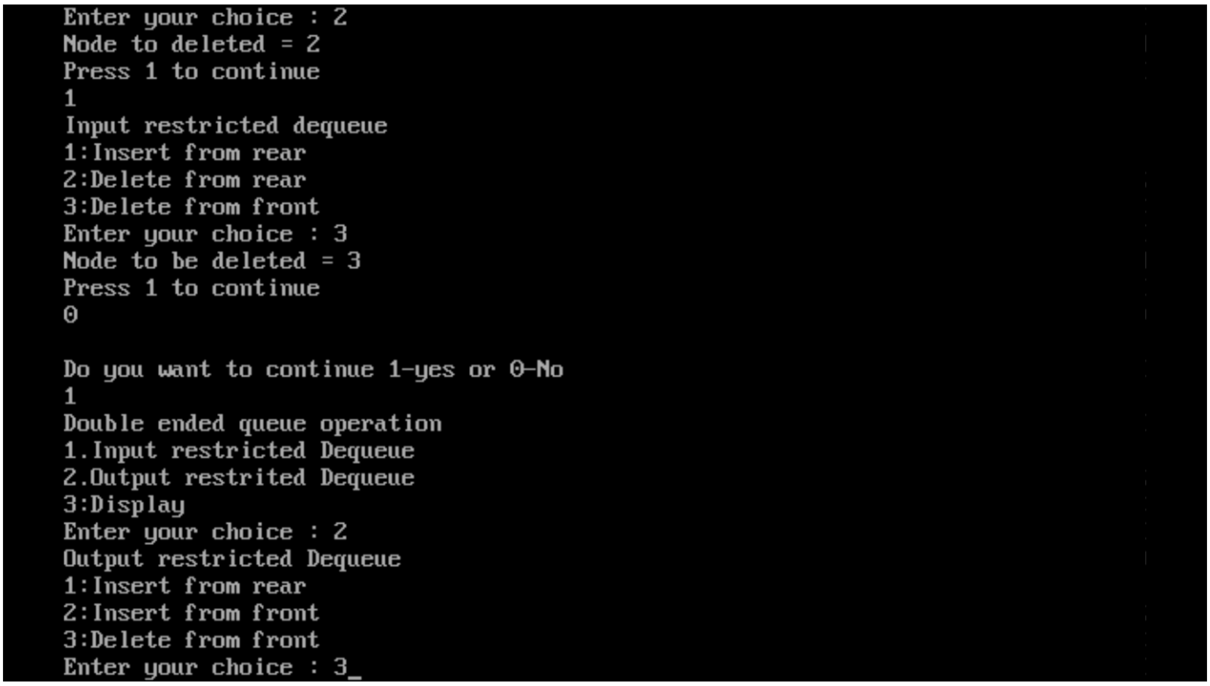
```
Double ended queue operation
1.Input restricted Dequeue
2.Output restrited Dequeue
3:Display
Enter your choice : 1
Input restricted dequeue
1:Insert from rear
2:Delete from rear
3:Delete from front
Enter your choice : 1
Enter the data to be stored
1
Press 1 to continue
0

Do you want to continue 1-yes or 0-No
1
Double ended queue operation
1.Input restricted Dequeue
2.Output restrited Dequeue
3:Display
Enter your choice : 2_
```



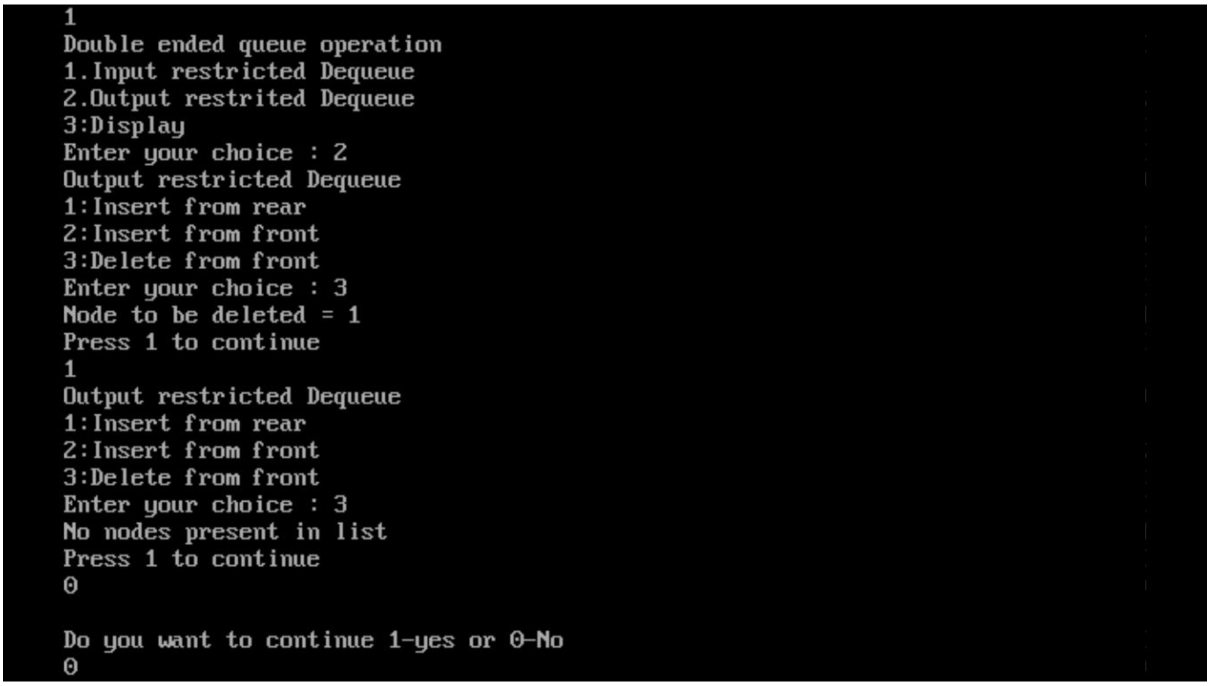
```
1.Input restricted Dequeue
2.Output restrited Dequeue
3:Display
Enter your choice : 2
Output restricted Dequeue
1:Insert from rear
2:Insert from front
3:Delete from front
Enter your choice : 1
Enter the data to be stored
2
Press 1 to continue
1
Output restricted Dequeue
1:Insert from rear
2:Insert from front
3:Delete from front
Enter your choice : 2
Enter the data to be stored
3
Press 1 to continue
0

Do you want to continue 1-yes or 0-No
1
```



```
0

Do you want to continue 1-yes or 0-No
1
Double ended queue operation
1.Input restricted Dequeue
2.Output restrited Dequeue
3:Display
Enter your choice : 3
The contents of the list are
3       1       2

Do you want to continue 1-yes or 0-No
1
Double ended queue operation
1.Input restricted Dequeue
2.Output restrited Dequeue
3:Display
Enter your choice :
1
Input restricted dequeue
1:Insert from rear
2:Delete from rear
3:Delete from front
Enter your choice : 2
```



```
Enter your choice : 2
Node to deleted = 2
Press 1 to continue
1
Input restricted dequeue
1:Insert from rear
2:Delete from rear
3:Delete from front
Enter your choice : 3
Node to be deleted = 3
Press 1 to continue
0

Do you want to continue 1-yes or 0-No
1
Double ended queue operation
1.Input restricted Dequeue
2.Output restrited Dequeue
3:Display
Enter your choice : 2
Output restricted Dequeue
1:Insert from rear
2:Insert from front
3:Delete from front
Enter your choice : 3_
```



```
1
Double ended queue operation
1.Input restricted Dequeue
2.Output restrited Dequeue
3:Display
Enter your choice : 2
Output restricted Dequeue
1:Insert from rear
2:Insert from front
3:Delete from front
Enter your choice : 3
Node to be deleted = 1
Press 1 to continue
1
Output restricted Dequeue
1:Insert from rear
2:Insert from front
3:Delete from front
Enter your choice : 3
No nodes present in list
Press 1 to continue
0

Do you want to continue 1-yes or 0-No
0
```

6. Various operations on doubly linked list.

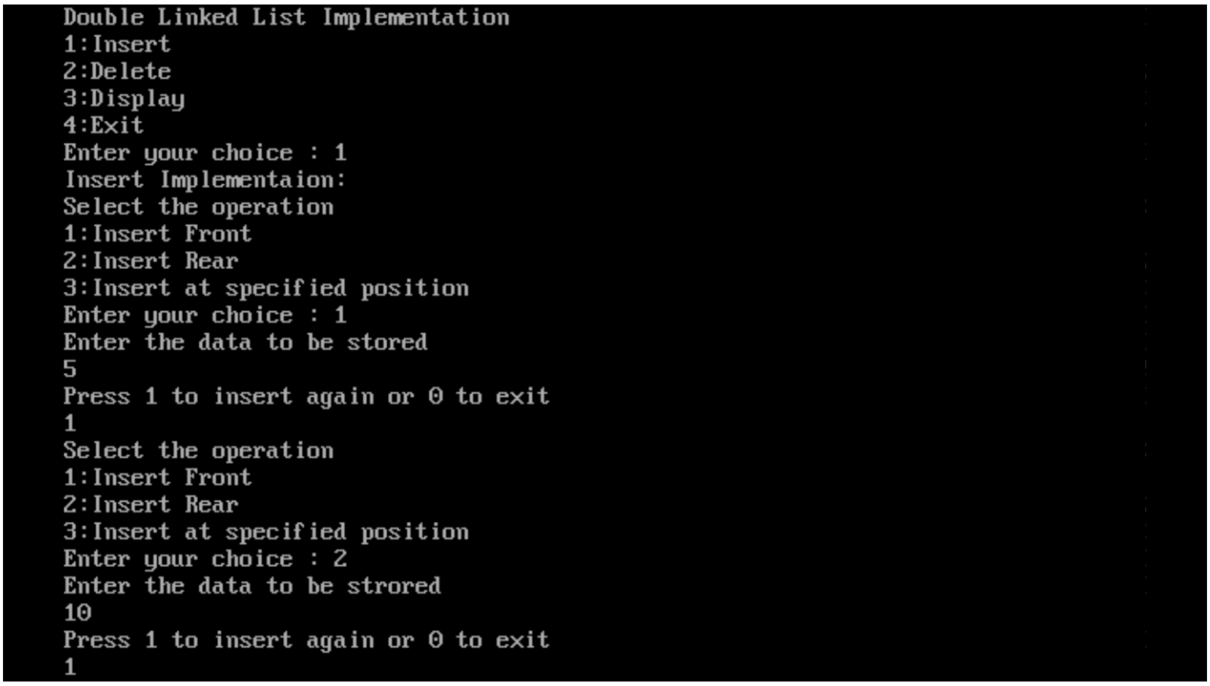
```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
        int data;
        struct node *llink;
        struct node *rlink;
};
typedef struct node *NODE;
NODE first=0;
int i,choice,ch1=1,choice1,ch2=1,choice2,ch3=1;
NODE insert_front(NODE);
NODE insert_rear(NODE);
NODE insert_pos(NODE);
NODE delete_front(NODE);
NODE delete_rear(NODE);
NODE delete_pos(NODE);
void display(NODE);
int main()
{
        int ch1=1;
        clrscr();
        while(ch1==1)
        {
                printf("Double Linked List Implementation\n");
                printf("1:Insert\n2:Delete\n3:Display\n4:Exit\n");
                printf("Enter your choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: printf("Insert Implementaion:\n");
                ch2=1;
                while(ch2==1)
                {
                    printf("Select the operation\n");
                    printf("1:Insert Front\n2:Insert Rear\n3:Insert at 
specified position\n");
                    printf("Enter your choice : ");
                    scanf("%d",&choice1);
                    switch(choice1)
                    {
                        case 1: first=insert_front(first);
                            break;
                        case 2: first=insert_rear(first);
                             break;
                        case 3: first=insert_pos(first);
                            break;
                        default: printf("Wrong choice\n");
                            break;
                    }
                    printf("Press 1 to insert again or 0 to exit\n");
                    scanf("%d",&ch2);
                }
                break;
            case 2: printf("Delete Implementation:\n");
                ch3=1;
                while(ch3==1)
                {
                    printf("Selete the operation\n");
                    printf("1:Delete Front\n2:Delete Rear\n3:Delete at
specified position\n");
                    printf("Enter your choice : ");
                    scanf("%d",&choice2);
                    switch(choice2)
                    {
                        case 1: first=delete_front(first);
                             break;
                        case 2: first=delete_rear(first);
                             break;
                        case 3: first=delete_pos(first);
                            break;
                        default: printf("Wrong choice\n");
                             break;
                    }
                    printf("Press 1 to delete again or 0 to exit\n");
                    scanf("%d",&ch3);
                }
                break;
            case 3:display(first);
                break;
            case 4:exit(0);
                break;
            default :printf("Wrong choice\n");
                break;
        }
        printf("\nDo you want to continue 1-yes or 0-No\n");
        scanf("%d",&ch1);
        }
        return 0;
}

NODE insert_front(NODE first)
{
        NODE newnode;
        newnode=(NODE)malloc(sizeof(struct node));
        newnode->llink=newnode->rlink=0;
        printf("Enter the data to be stored\n");
        scanf("%d",&newnode->data);
        if(first==0)
        {
                first=newnode;
                return(first);
        }
        newnode->rlink=first;
        first->llink=newnode;
        first=newnode;
        return(first);
}

NODE insert_rear(NODE first)
{
        NODE newnode,temp;
        newnode=(NODE)malloc(sizeof(struct node));
        newnode->rlink=newnode->llink=0;
        printf("Enter the data to be strored\n");
        scanf("%d",&newnode->data);
```

```
	if(first==0)
	{
		first=newnode;
		return(first);
	}
	temp=first;
	while(temp->rlink!=0)
		temp=temp->rlink;
	temp->rlink=newnode;
	newnode->llink=temp;
	return(first);
}

NODE delete_front(NODE first)
{
	NODE temp;
	if(first==0)
	{
		printf("No nodes in the list\n");
		return(first);
	}
	temp=first;
	if(temp->rlink==0)
	{
		printf("Deleted node is = %d\n",temp->data);
		free(temp);
		return(first);
	}
	printf("Deleted node is = %d\n",temp->data);
```

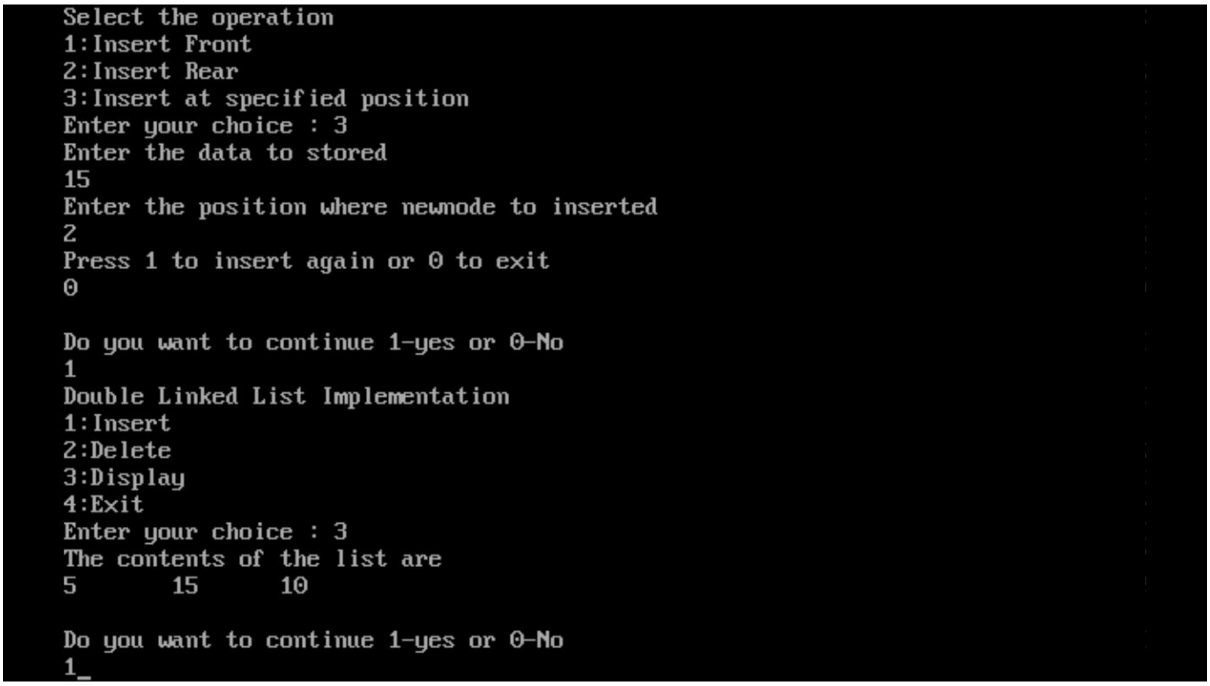
```
        first=first->rlink;
        first->llink=0;
        free(temp);
        return(first);
}

NODE delete_rear(NODE first)
{
        NODE temp,temp1;
        if(first==0)
        {
                printf("No nodes in the list\n");
                return(first);
        }
        temp=first;
        if(first->rlink==0)
        {
                printf("Deleted node is = %d\n",temp->data);
                free(temp);
                first=0;
                return(first);
        }
        while(temp->rlink!=0)
                temp=temp->rlink;
        printf("Deleted node is = %d\n",temp->data);
        temp1=temp->llink;
        free(temp);
        temp1->rlink=0;
        return(first);
```

```
}

NODE insert_pos(NODE first)
{
        NODE newnode,temp,temp1;
        int pos;
        newnode=(NODE)malloc(sizeof(struct node));
        newnode->rlink=newnode->llink=0;
        printf("Enter the data to stored\n");
        scanf("%d",&newnode->data);
        printf("Enter the position where newnode to inserted\n");
        scanf("%d",&pos);
        temp=first;
        if((pos==1)&&(first==0))
        {
                first=newnode;
                return(first);
        }
        if(pos==1)
        {
                newnode->rlink=first;
                first->llink=newnode;
                first=newnode;
                return(first);
        }
        for(i=1;i<pos;i++)
        {
                temp1=temp;
                temp=temp->rlink;
        }
        if(temp==0)
        {
                printf("Invalid position\n");
                return(first);
        }
        temp1->rlink=newnode;
        newnode->llink=temp1;
        newnode->rlink=temp;
        temp->llink=newnode;
        return(first);
}

NODE delete_pos(NODE first)
{
        NODE temp,temp1,temp2;
        int pos;
        if(first==0)
        {
                printf("No nodes in the list\n");
                return(first);
        }
        printf("Enter the position where node to deleted\n");
        scanf("%d",&pos);
        temp=first;
        if((pos==1)&&(first->rlink==0))
        {
                printf("Deleted data = %d\n",temp->data);
                free(temp);
                first=0;
                return(first);
        }
        if(pos==1)
        {
                printf("Deleted data = %d\n",temp->data);
                first=first->rlink;
                first->llink=0;
                free(temp);
                return(first);
        }
        for(i=1;i<pos;i++)
                temp=temp->rlink;
        if(temp==0)
        {
                printf("Invalid position\n");
                return(first);
        }
        temp1=temp->llink;
        temp2=temp->rlink;
        printf("Deleted data = %d\n",temp->data);
        temp1->rlink=temp2;
        temp2->llink=temp1;
        free(temp);
        return(first);
}

void display(NODE first)
{
        NODE temp;
        temp=first;
        if(first==0)
                printf("No nodes are present in list\n");
        else
        {
                printf("The contents of the list are\n");
                for(temp=first;temp!=0;temp=temp->rlink)
                        printf("%d\t",temp->data);
        }
        printf("\n");
}
```

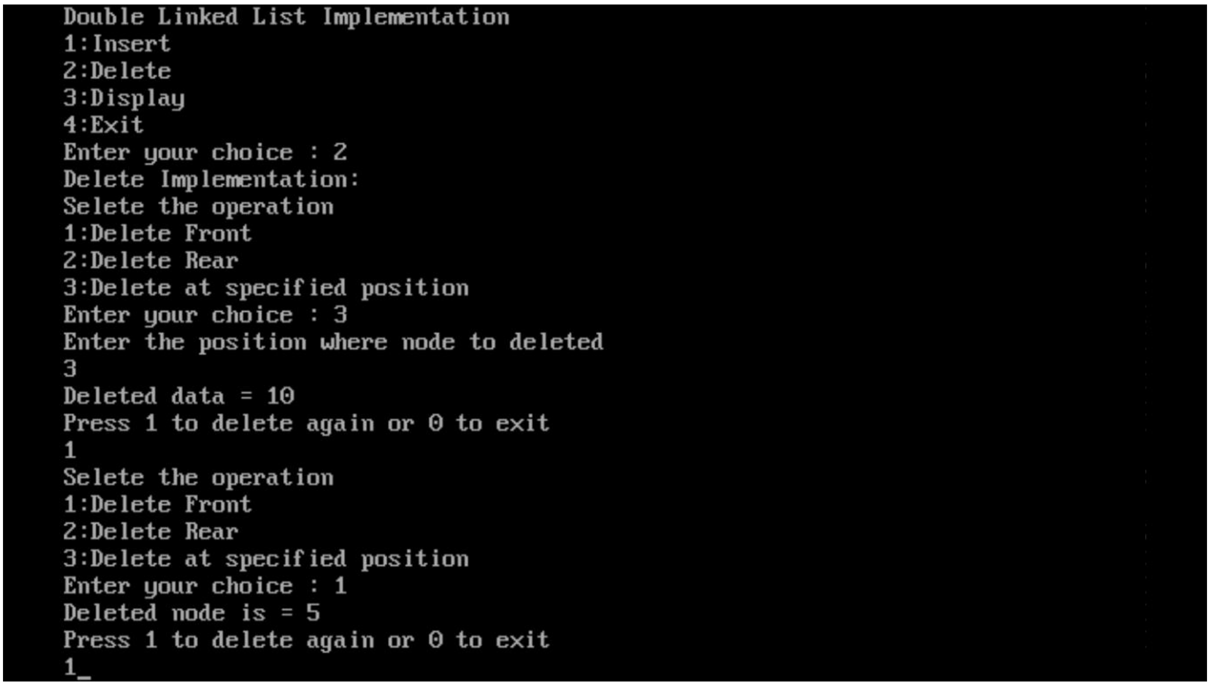
OUTPUT:



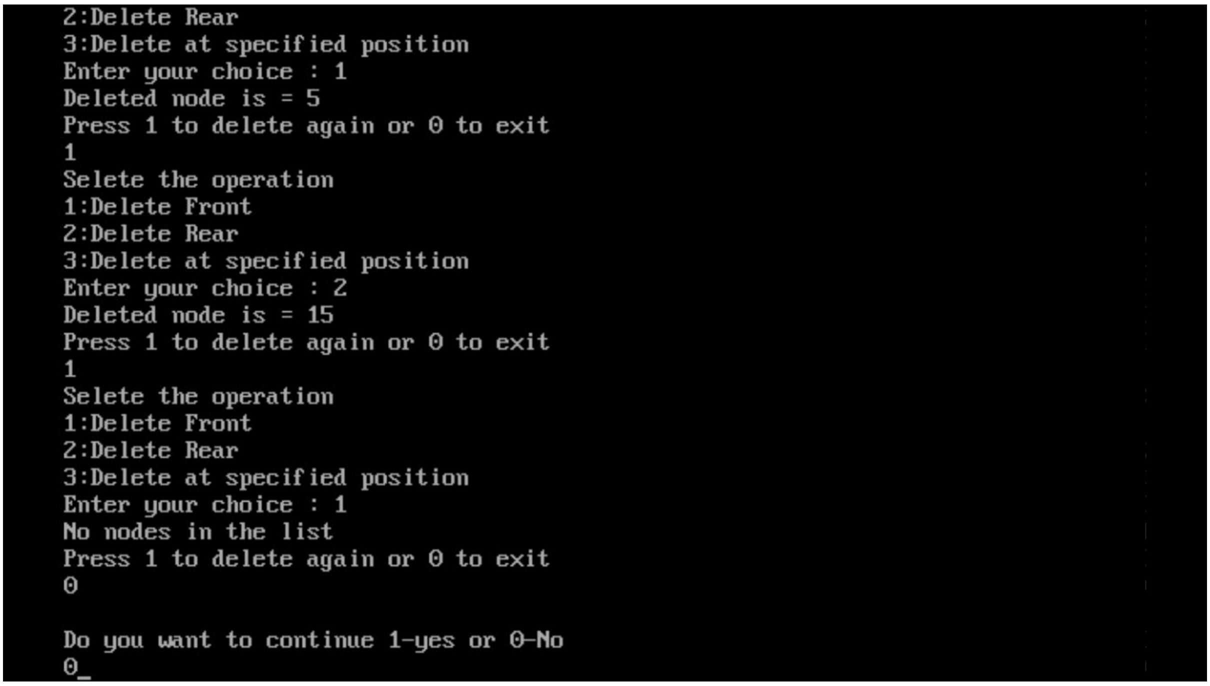
```
Double Linked List Implementation
1:Insert
2:Delete
3:Display
4:Exit
Enter your choice : 1
Insert Implementaion:
Select the operation
1:Insert Front
2:Insert Rear
3:Insert at specified position
Enter your choice : 1
Enter the data to be stored
5
Press 1 to insert again or 0 to exit
1
Select the operation
1:Insert Front
2:Insert Rear
3:Insert at specified position
Enter your choice : 2
Enter the data to be strored
10
Press 1 to insert again or 0 to exit
1
```



```
Select the operation
1:Insert Front
2:Insert Rear
3:Insert at specified position
Enter your choice : 3
Enter the data to stored
15
Enter the position where newnode to inserted
2
Press 1 to insert again or 0 to exit
0

Do you want to continue 1-yes or 0-No
1
Double Linked List Implementation
1:Insert
2:Delete
3:Display
4:Exit
Enter your choice : 3
The contents of the list are
5       15      10

Do you want to continue 1-yes or 0-No
1_
```



```
Double Linked List Implementation
1:Insert
2:Delete
3:Display
4:Exit
Enter your choice : 2
Delete Implementation:
Selete the operation
1:Delete Front
2:Delete Rear
3:Delete at specified position
Enter your choice : 3
Enter the position where node to deleted
3
Deleted data = 10
Press 1 to delete again or 0 to exit
1
Selete the operation
1:Delete Front
2:Delete Rear
3:Delete at specified position
Enter your choice : 1
Deleted node is = 5
Press 1 to delete again or 0 to exit
1_
```



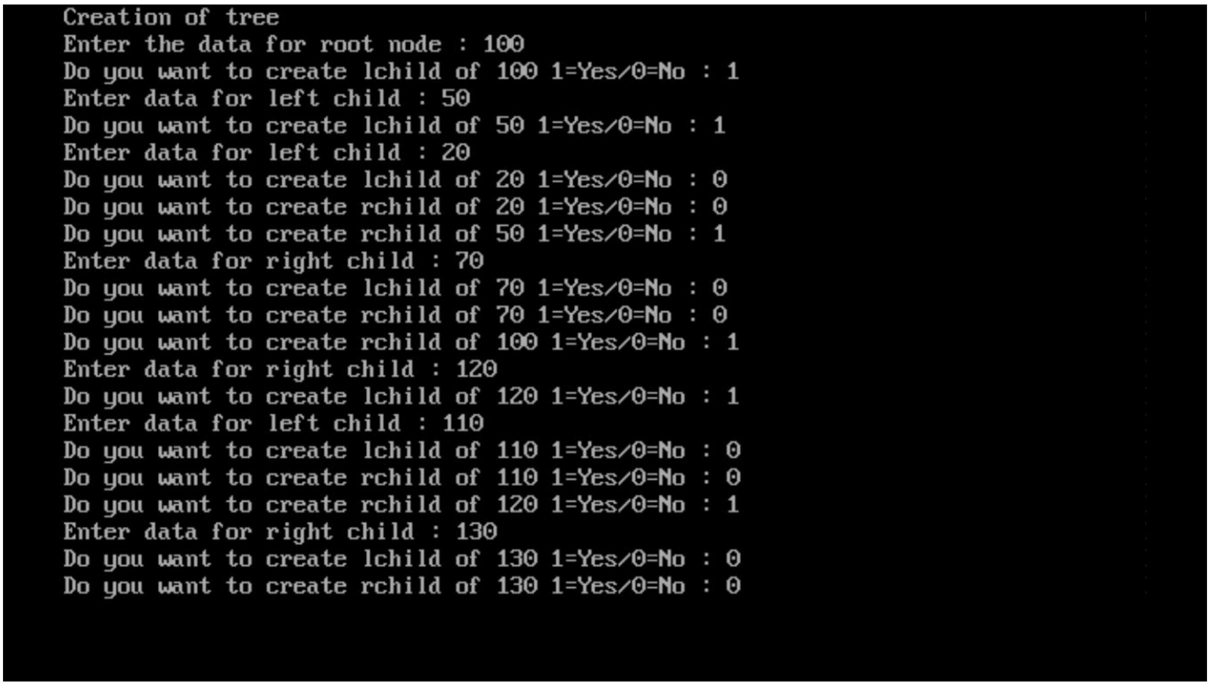
```
2:Delete Rear
3:Delete at specified position
Enter your choice : 1
Deleted node is = 5
Press 1 to delete again or 0 to exit
1
Selete the operation
1:Delete Front
2:Delete Rear
3:Delete at specified position
Enter your choice : 2
Deleted node is = 15
Press 1 to delete again or 0 to exit
1
Selete the operation
1:Delete Front
2:Delete Rear
3:Delete at specified position
Enter your choice : 1
No nodes in the list
Press 1 to delete again or 0 to exit
0

Do you want to continue 1-yes or 0-No
0_
```

7. Binary tree and traverse in order, preorder and postorder.

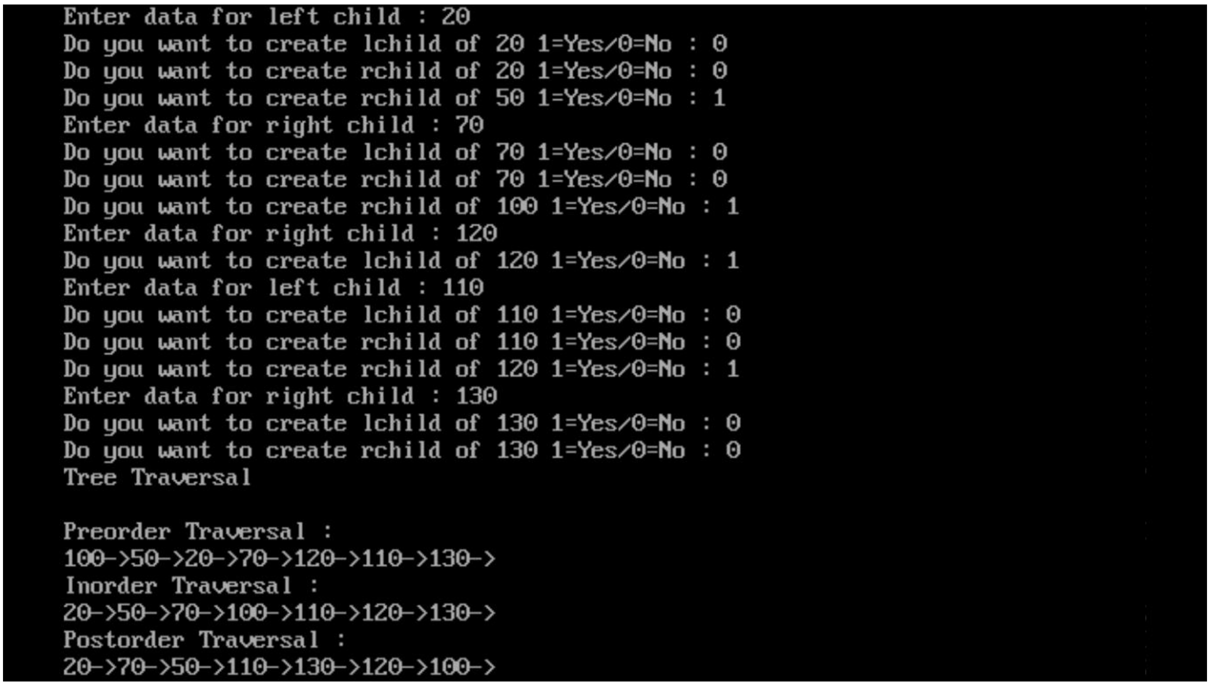
```
#include<stdio.h>
#include<conio.h>
struct node
{
	int data;
	struct node *lchild;
	struct node *rchild;
};
typedef struct node *NODE;
NODE root=0;
void create(NODE *);
void inorder(NODE);
void preorder(NODE);
void postorder(NODE);
int is_lchild(NODE *);
int is_rchild(NODE *);
void main()
{
	clrscr();
	printf("Creation of tree\n");
	root=(NODE)malloc(sizeof(struct node));
	printf("Enter the data for root node : ");
	scanf("%d",&root->data);
	create(&root);
	printf("Tree Traversal\n");
	printf("\nPreorder Traversal :\n");
	preorder(root);
	printf("\nInorder Traversal :\n");
```

```
        inorder(root);
        printf("\nPostorder Traversal :\n");
        postorder(root);
        getch();
}

void create(NODE *root1)
{
        NODE temp,temp1;
        if(is_lchild(&(*root1)))
        {
                (*root1)->lchild=(NODE)malloc(sizeof(struct node));
                temp=(*root1)->lchild;
                printf("Enter data for left child : ");
                scanf("%d",&temp->data);
                create(&temp);
        }
        else
                (*root1)->lchild=0;
        if(is_rchild(&(*root1)))
        {
                (*root1)->rchild=(NODE)malloc(sizeof(struct node));
                temp1=(*root1)->rchild;
                printf("Enter data for right child : ");
                scanf("%d",&temp1->data);
                create(&temp1);
        }
        else
                (*root1)->rchild=0;
        return;
}

int is_lchild(NODE *root2)
{
        int ch;
        printf("Do you want to create lchild of %d 1=Yes/0=No : ",(*root2)->data);
        scanf("%d",&ch);
        if(ch==1)
                return(1);
        else
                return(0);
}

int is_rchild(NODE *root2)
{
        int ch1;
        printf("Do you want to create rchild of %d 1=Yes/0=No : ",(*root2)->data);
        scanf("%d",&ch1);
        if(ch1==1)
                return(1);
        else
                return(0);
}

void preorder(NODE root4)
{
        if(root4!=0)
        {
                printf("%d->",root4->data);
                preorder(root4->lchild);
                preorder(root4->rchild);
        }
}

void inorder(NODE root5)
{
        if(root5!=0)
        {
                inorder(root5->lchild);
                printf("%d->",root5->data);
                inorder(root5->rchild);
        }
}

void postorder(NODE root6)
{
        if(root6!=0)
        {
                postorder(root6->lchild);
                postorder(root6->rchild);
                printf("%d->",root6->data);
        }
}
```

OUTPUT:



```
Creation of tree
Enter the data for root node : 100
Do you want to create lchild of 100 1=Yes/0=No : 1
Enter data for left child : 50
Do you want to create lchild of 50 1=Yes/0=No : 1
Enter data for left child : 20
Do you want to create lchild of 20 1=Yes/0=No : 0
Do you want to create rchild of 20 1=Yes/0=No : 0
Do you want to create rchild of 50 1=Yes/0=No : 1
Enter data for right child : 70
Do you want to create lchild of 70 1=Yes/0=No : 0
Do you want to create rchild of 70 1=Yes/0=No : 0
Do you want to create rchild of 100 1=Yes/0=No : 1
Enter data for right child : 120
Do you want to create lchild of 120 1=Yes/0=No : 1
Enter data for left child : 110
Do you want to create lchild of 110 1=Yes/0=No : 0
Do you want to create rchild of 110 1=Yes/0=No : 0
Do you want to create rchild of 120 1=Yes/0=No : 1
Enter data for right child : 130
Do you want to create lchild of 130 1=Yes/0=No : 0
Do you want to create rchild of 130 1=Yes/0=No : 0
```

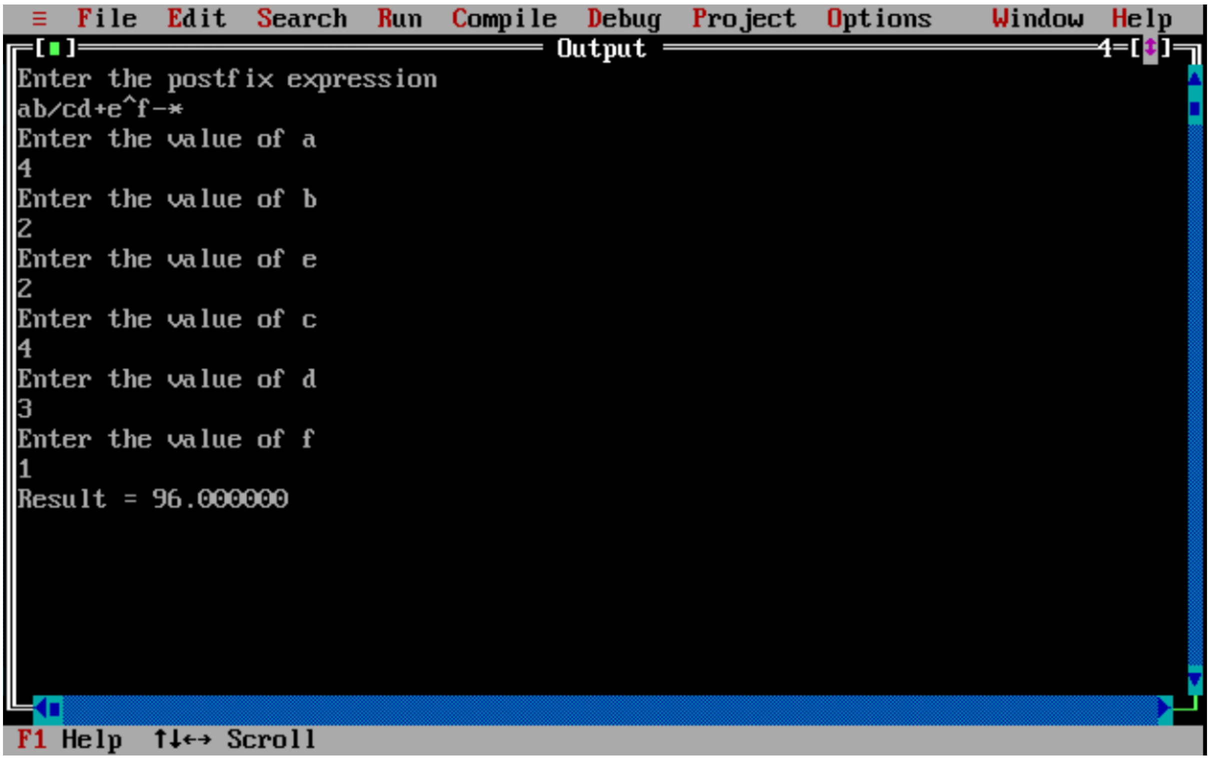


```
Enter data for left child : 20
Do you want to create lchild of 20 1=Yes/0=No : 0
Do you want to create rchild of 20 1=Yes/0=No : 0
Do you want to create rchild of 50 1=Yes/0=No : 1
Enter data for right child : 70
Do you want to create lchild of 70 1=Yes/0=No : 0
Do you want to create rchild of 70 1=Yes/0=No : 0
Do you want to create rchild of 100 1=Yes/0=No : 1
Enter data for right child : 120
Do you want to create lchild of 120 1=Yes/0=No : 1
Enter data for left child : 110
Do you want to create lchild of 110 1=Yes/0=No : 0
Do you want to create rchild of 110 1=Yes/0=No : 0
Do you want to create rchild of 120 1=Yes/0=No : 1
Enter data for right child : 130
Do you want to create lchild of 130 1=Yes/0=No : 0
Do you want to create rchild of 130 1=Yes/0=No : 0
Tree Traversal

Preorder Traversal :
100->50->20->70->120->110->130->
Inorder Traversal :
20->50->70->100->110->120->130->
Postorder Traversal :
20->70->50->110->130->120->100->
```

8. Evaluation expression tree using binary tree.

```
#include<stdio.h>

#include<conio.h>

#include<math.h>

struct node

{

        int data;

        struct node *lchild;

        struct node *rchild;

};

typedef struct node *NODE;

NODE root=0;

NODE create_tree(char postfix[]);

float eval(NODE root);

void main()

{

        char postfix[20];

        float result;

        clrscr();

        printf("Enter the postfix expression\n");

        scanf("%s",postfix);

        root=create_tree(postfix);

        result=eval(root);

        printf("Result = %f\n",result);

        getch();

}


NODE create_tree(char postfix[])

{
        NODE temp,stack[20];
        int i=0,j=0;
        char symbol;
        for(i=0;(symbol=postfix[i])!='\0';i++)
        {
                temp=(NODE)malloc(sizeof(struct node));
                temp->lchild=0;
                temp->rchild=0;
                temp->data=symbol;
                if(isalnum(symbol))
                        stack[j++]=temp;
                else
                {
                        temp->rchild=stack[--j];
                        temp->lchild=stack[--j];
                        stack[j++]=temp;
                }
        }
        return(stack[--j]);
}

float eval(NODE root)
{
        float num;
        switch(root->data)
        {
                case '+' : return eval(root->lchild)+eval(root->rchild);
                case '-' : return eval(root->lchild)-eval(root->rchild);
                case '/' : return eval(root->lchild)/eval(root->rchild);
                case '*' : return eval(root->lchild)*eval(root->rchild);
                case '^' : return pow(eval(root->lchild),eval(root->rchild));
                default : if(isalpha(root->data))
                          {
                                printf("Enter the value of %c\n",root->data);
                                scanf("%f",&num);
                                return(num);
                          }
                          else
                                return(root->data-'\0');
        }
}
```

OUTPUT:


≡ File Edit Search Run Compile Debug Project Options Window Help
[■]
Output
4=[↕]
F1 Help ↑↓←→ Scroll

```
Enter the postfix expression
ab/cd+e^f-*
Enter the value of a
4
Enter the value of b
2
Enter the value of e
2
Enter the value of c
4
Enter the value of d
3
Enter the value of f
1
Result = 96.000000
```

9. Perform Insert and Delete operations in binary search tree.

```
#include<stdio.h>

#include<conio.h>

#define TRUE 1

#define FALSE 0

struct node

{

        int data;

        struct node *lchild;

        struct node *rchild;

};

typedef struct node *NODE;

int i=1,num,req;

void insert(NODE*,int) ;

void node_delete(NODE*);

search(NODE*,int,NODE*,NODE*,int*);

void inorder(NODE root4);

void main()

{

        NODE root=0;

        clrscr();

        printf("Enter the num of nodes\n");

        scanf("%d",&req);

        while(i++<=req)

        {

                printf("Enter the data : ");

                scanf("%d",&num);

                insert(&root,num);

        }
        printf("\nNodes before deletion\n");
        inorder(root);
        node_delete(&root);
        printf("\nNodes after deletion\n");
        inorder(root);
        getch();
}

void insert(NODE *(root1),int num)
{
        if((*root1)==0)
        {
                (*root1)=(NODE)malloc(sizeof(struct node));
                (*root1)->lchild=(*root1)->rchild=0;
                (*root1)->data=num;
        }
        else
        {
                if(num<((*root1)->data))
                        insert(&((*root1)->lchild),num);
                else
                        insert(&((*root1)->rchild),num);
        }
        return;
}

void inorder(NODE root4)
{
        if(root4!=0)
        {
                inorder(root4->lchild);
                printf("%d->",root4->data);
                inorder(root4->rchild);
        }
}

search(NODE *root3,int num2,NODE *par,NODE *x,int *found)
{
        NODE q;
        q=*root3;
        *found=FALSE;
        *par=0;
        while(q!=0)
        {
                if(num2==q->data)
                {
                        *found=TRUE;
                        *x=q; return;
                }
                *par=q;
                if(num2<q->data)
                        q=q->lchild;
                else
                        q=q->rchild;
        }
        return;
}
void node_delete(NODE *root2)
```

```
{
        int num,found;
        NODE parent,x,xsucc;
        parent=0;x=0;
        if(*root2==0)
        {
                printf("Tree is empty\n");
                return;
        }
        printf("\n");
        printf("\nEnter data to be deleted\n");
        scanf("%d",&num);
        search(&(*root2),num,&parent,&x,&found);
        if(found==FALSE)
        {
                printf("Data to be deleted %d is not found\n",num);
                return;
        }
        printf("Data to be deleted %d is found\n",num);
        if(x->lchild!=0&&x->rchild!=0)
        {
                parent=x;
                xsucc=x->rchild;
                while(xsucc->lchild!=0)
                {
                        parent=xsucc;
                        xsucc=x->lchild;
                }
                x->data=xsucc->data;
                x=xsucc;
        }
        if(x->lchild!=0&&x->rchild!=0)
        {
                if(parent->lchild==x)
                        parent->lchild=x->lchild;
                else
                        parent->rchild=x->lchild;
                free(x);
                return;
        }
        if(x->rchild!=0&&x->lchild==0)
        {
                if(parent->lchild==x)
                        parent->lchild=x->rchild;
                else
                        parent->rchild=x->rchild;
                free(x);
                return;
        }
        if(x->lchild==0&&x->rchild==0)
        {
                if(parent->rchild==x)
                        parent->lchild=0;
                else
                        parent->lchild=0;
                free(x);
                return;
        }
```

```
}
```

OUTPUT:

10. Right-in-threaded binary tree.

```
#include<stdio.h>
#include<conio.h>
struct node
{
        int data;
        struct node *left;
        struct node *right;
        int RT;
};
typedef struct node *NODE;
NODE head=0;
NODE create(int,NODE);
void insert_left(int,NODE);
void insert_right(int,NODE);
void inorder(NODE);
NODE inorder_successor(NODE);
int ch,i,n,item,choice;

void main()
{
        NODE head=0;
        clrscr();
        head=(NODE)malloc(sizeof(struct node));
        head->right=head;
        head->left=0;
        head->RT=0;
        while(1)
        {
                printf("1:Create tree\n2:Inorder\n3.Exit\n");
                printf("Enter the choice : ");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1: printf("Enter number of nodes to create : \n");
                                scanf("%d",&n);
                                for(i=1;i<n+1;i++)
                                {
                                        printf("Enter %d data : ",i);
                                        scanf("%d",&item);
                                        head=create(item,head);
                                }
                                break;
                        case 2: inorder(head);
                                break;
                        case 3: exit(0);
                        default :printf("Wrong choice\n");
                                        break;
                }
        printf("\n");
        }
}

NODE create(int item,NODE head)
{
        NODE curptr,ptr;
        if(head->left==0)
```

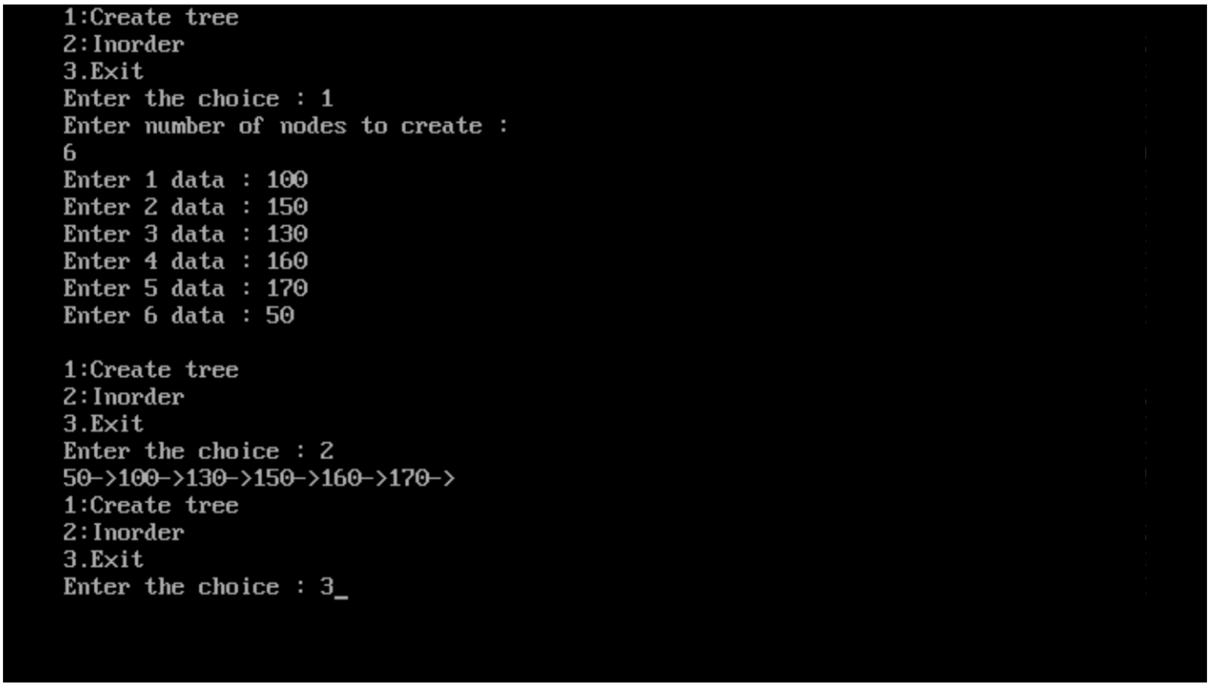
```
{
        insert_left(item,head);
        return(head);
}
curptr=head->left;
while(curptr!=head)
{
        ptr=curptr;
        if(item<(curptr->data))
        {
                if(curptr->left!=0)
                        curptr=curptr->left;
                else
                        break;
        }
        else
        {
                if(item>(curptr->data))
                {
                        if(curptr->RT==0)
                                curptr=curptr->right;
                        else
                                break;
                }
        }
}
if(item<(curptr->data))
{
        insert_left(item,ptr);
```

```
                        return(head);
            }
            else
            {
                        if(item>(curptr->data)&&curptr->RT==1)
                        insert_right(item,ptr);
            }
            return(head);
}

void insert_left(int item,NODE ptr)
{
            NODE temp,newnode;
            newnode=(NODE)malloc(sizeof(struct node));
            newnode->left=0;
            newnode->data=item;
            ptr->left=newnode;
            newnode->right=ptr;
            newnode->RT=1;
}

void insert_right(int item,NODE ptr)
{
            NODE temp,newnode;
            newnode=(NODE)malloc(sizeof(struct node));
            newnode->left=0;
            newnode->data=item;
            temp=ptr->right;
            ptr->right=newnode;
```

```
        ptr->RT=0;
        newnode->right=temp;
        newnode->RT=1;
}

void inorder(NODE head)
{
        NODE temp;
        if(head->left==0)
        {
                printf("\n No node in the tree");
                return;
        }
        temp=head;
        while(1)
        {
                temp=inorder_successor(temp);
                if(temp==head)
                        return;
                printf("%d->",temp->data);
        }
        printf("\n");
}

NODE inorder_successor(NODE ptr)
{
        NODE temp;
        temp=ptr->right;
        if(ptr->RT==1)
```

```
                return(temp);
        while(temp->left!=0)
                temp=temp->left;
        return(temp);
}
```

OUTPUT:



```
1:Create tree
2:Inorder
3.Exit
Enter the choice : 1
Enter number of nodes to create :
6
Enter 1 data : 100
Enter 2 data : 150
Enter 3 data : 130
Enter 4 data : 160
Enter 5 data : 170
Enter 6 data : 50

1:Create tree
2:Inorder
3.Exit
Enter the choice : 2
50->100->130->150->160->170->
1:Create tree
2:Inorder
3.Exit
Enter the choice : 3_
```