DATA STRUCTURES LAB PROGRAMS

1BM19CS085

MD IBADUDDIN SAFFAN

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

CODE:

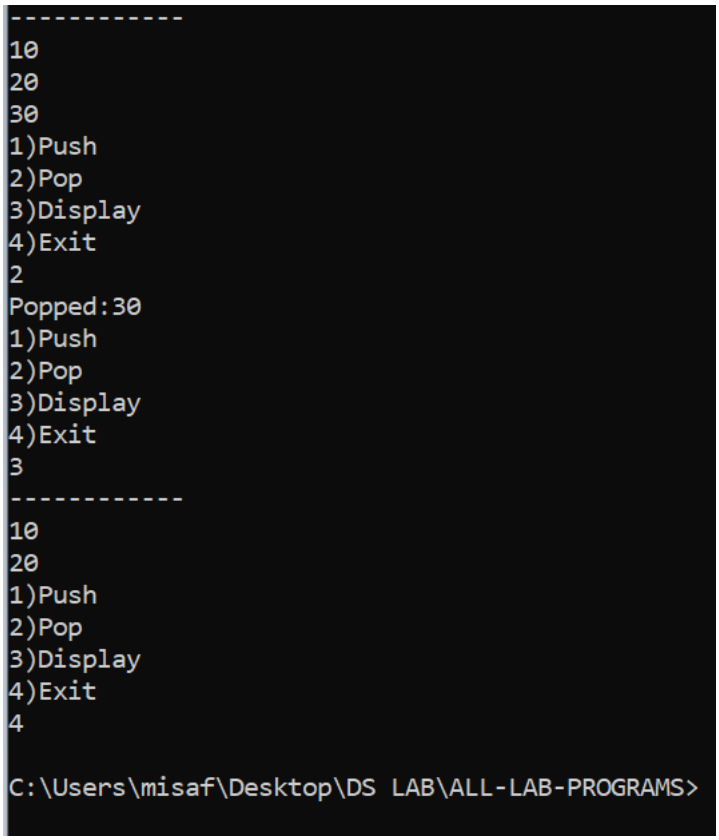
```
#include<stdio.h>
int size,s[100],item,front=0,top=-1;

void push()
{
	if(top==size-1)
	{
		printf("STACK OVERFLOW!\n");return;
	}
	top++;
	s[top]=item;
}
int pop()
{
	if(top<front)
	{
		printf("STACK UNDERFLOW!\n");return -1;
	}
	top--;
	return s[top+1];
}
void display()
{
	printf("----------\n");
	for(int i=front;i<=top;i++)
	{
		printf("%d\n",s[i]);
	}
}
```

```
void main()
{
	printf("Enter size:\n");
	scanf("%d",&size);
	int c=1,ch,pop_item;
	while(c==1)
	{
		printf("1)Push\n2)Pop\n3)Display\n4)Exit\n");
		scanf("%d",&ch);
		switch(ch)
		{
			case 1:
				printf("Enter item:\n");
				scanf("%d",&item);push();break;
			case 2:
				pop_item=pop();
				if(pop_item!=-1)
				printf("Popped:%d\n",pop_item);
				break;
			case 3:display();break;
			case 4:c=0;break;
			default:printf("Invalid choice!\n");
		}

	}
}
```

OUTPUT:

```
C:\Users\misaf\Desktop\DS LAB\ALL-LAB-PROGRAMS>lab1c
Enter size:
3
1)Push
2)Pop
3)Display
4)Exit
2
STACK UNDERFLOW!
1)Push
2)Pop
3)Display
4)Exit
1
Enter item:
10
1)Push
2)Pop
3)Display
4)Exit
1
Enter item:
20
1)Push
2)Pop
3)Display
4)Exit
1
```

```
Enter item:
20
1)Push
2)Pop
3)Display
4)Exit
1
Enter item:
30
1)Push
2)Pop
3)Display
4)Exit
1
Enter item:
40
STACK OVERFLOW!
1)Push
2)Pop
3)Display
4)Exit
3
------------
10
20
30
1)Push
2)Pop
3)Display
4)Exit
```



---

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)

CODE:

```
#include <stdio.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, j, i;
        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';
        printf("postfix expression:\n%s", postfix);
}
int main()
{
        char infix[20], postfix[20];
        printf("Enter the infix expression:\n");
        scanf("%s", &infix);
        infix_postfix(infix, postfix);
        return 0;
}
```

OUTPUT:

```
C:\Users\misaf\Desktop\DS LAB\ALL-LAB-PROGRAMS>lab2c
Enter the infix expression:
a+b*(c^d-e)^(f+g*h)-i
postfix expression:
abcd^e-fgh*+^*+i-
```

---

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

CODE:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define SIZE 3
int item, front = 0, rear = -1, q[10];
void insertrear()
{
	if (rear == 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 display()
{
	int i;
	if (front > rear)
	{
		printf("Queue is EMPTY!!\n");
		return;
```

```
        }
        printf("Contents of Queue:\n----------\n");
        for (i = front; i <= rear; i++)
        {
                printf("%d\n", q[i]);
        }
}
void main()
{
        int choice;
        while (1)
        {
                printf("\n1 : INSERT \n2 : DELETE\n3 : DISPLAY\n4 : EXIT\n");
                printf("Enter choice:\n");
                scanf("%d", &choice);
                switch (choice)
                {
                case 1:
                        printf("Enter 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:
                        display();
                        break;
                default:
                        exit(0);
                }
        }
}
```

OUTPUT:

```
C:\Users\misaf\Desktop\DS LAB\ALL-LAB-PROGRAMS>lab3c

1 : INSERT
2 : DELETE
3 : DISPLAY
4 : EXIT
Enter choice:
2
Queue is empty

1 : INSERT
2 : DELETE
3 : DISPLAY
4 : EXIT
Enter choice:
1
Enter item to be inserted:
10

1 : INSERT
2 : DELETE
3 : DISPLAY
4 : EXIT
Enter choice:
1
Enter item to be inserted:
20
```

```
1 : INSERT
2 : DELETE
3 : DISPLAY
4 : EXIT
Enter choice:
1
Enter item to be inserted:
30

1 : INSERT
2 : DELETE
3 : DISPLAY
4 : EXIT
Enter choice:
1
Enter item to be inserted:
40
Queue OVERFLOW!!

1 : INSERT
2 : DELETE
3 : DISPLAY
4 : EXIT
Enter choice:
3
Contents of Queue:
----------
10
20
30
```

```
1 : INSERT
2 : DELETE
3 : DISPLAY
4 : EXIT
Enter choice:
2
Item deleted : 10

1 : INSERT
2 : DELETE
3 : DISPLAY
4 : EXIT
Enter choice:
3
Contents of Queue:
----------
20
30

1 : INSERT
2 : DELETE
3 : DISPLAY
4 : EXIT
Enter choice:
4
```

---

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:

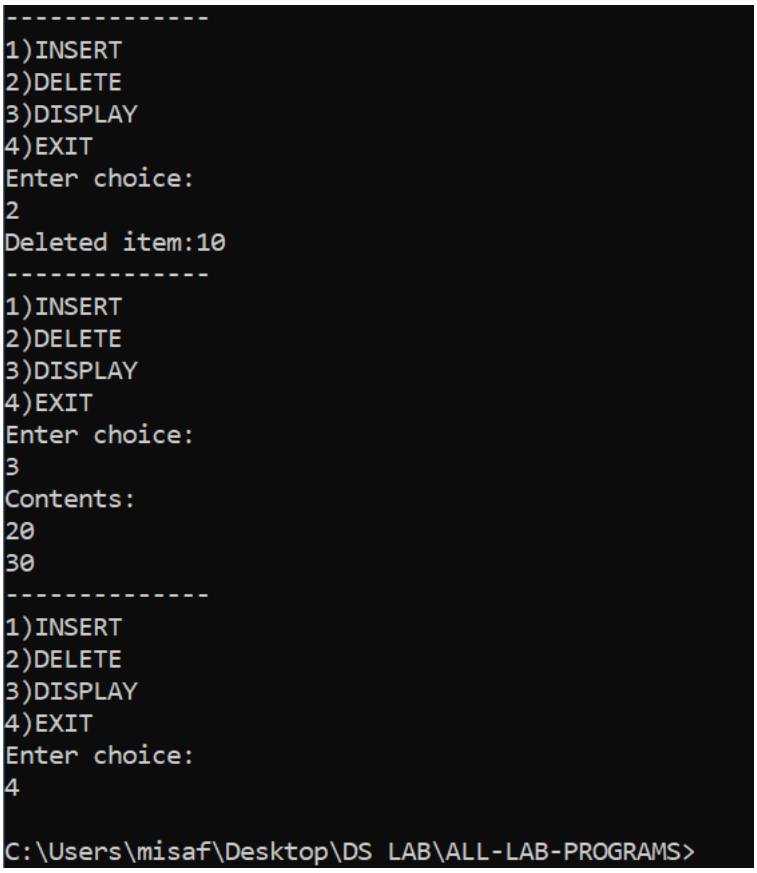
```
#include <stdio.h>
#define size 3
int item, f = 0, r = -1, q[size], count = 0;
void insertrear()
{
	if (count == size)
	{
		printf("OVERFLOW!!\n");
		return;
	}
	r = (r + 1) % size;
	q[r] = item;
	count++;
}
int deletefront()
{
	if (count == 0)
		return -1;
	item = q[f];
	f = (f + 1) % size;
	count = count - 1;
	return item;
}
void display()
{
	int i;
	if (count == 0)
	{
		printf("QUEUE IS EMPTY!\n");
		return;
	}
	int front = f;
	printf("Contents:\n");
	for (int i = 1; i <= count; i++)
	{
		printf("%d \n", q[front]);
		front = (front + 1) % size;
	}
}
void main()
{
```

```
    int choice, check = 1;
    while (check == 1)
    {
        printf("--------------\n1)INSERT\n2)DELETE\n3)DISPLAY\n4)EXIT\nEnter choice:\n");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            printf("Enter item:\n");
            scanf("%d", &item);
            insertrear();
            break;
        case 2:
            item = deletefront();
            if (item == -1)
                printf("QUEUE IS EMPTY!\n");
            else
                printf("Deleted item:%d\n", item);
            break;
        case 3:
            display();
            break;
        default:
            check = 0;
        }
    }
}
```

OUTPUT:

```
C:\Users\misaf\Desktop\DS LAB\ALL-LAB-PROGRAMS>lab4c
--------------
1)INSERT
2)DELETE
3)DISPLAY
4)EXIT
Enter choice:
2
QUEUE IS EMPTY!
--------------
1)INSERT
2)DELETE
3)DISPLAY
4)EXIT
Enter choice:
1
Enter item:
10
--------------
1)INSERT
2)DELETE
3)DISPLAY
4)EXIT
Enter choice:
1
Enter item:
20
--------------
```

```
--------------
1)INSERT
2)DELETE
3)DISPLAY
4)EXIT
Enter choice:
1
Enter item:
30
--------------
1)INSERT
2)DELETE
3)DISPLAY
4)EXIT
Enter choice:
1
Enter item:
40
OVERFLOW!!
--------------
1)INSERT
2)DELETE
3)DISPLAY
4)EXIT
Enter choice:
3
Contents:
10
20
30
```



---

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.

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("MEMORY FULL!\n");
                exit(0);
        }
        return x;
}
NODE insert_rear(NODE first,int item)
{
        NODE temp=getnode();
        temp->info=item;
        temp->link=NULL;
        NODE cur=first;
        if(first==NULL)return temp;
        while(cur->link!=NULL)
        {
                cur=cur->link;
        }
        cur->link=temp;
        return first;

}
NODE insert_front(NODE first,int item)
{
        NODE temp=getnode();
        temp->info=item;
        if(first==NULL)return temp;
        temp->link=first;
        return temp;
}
NODE insert_pos(NODE first,int item,int pos)
{
        NODE temp=getnode();
        temp->info=item;
```

```
        temp->link=NULL;
        NODE cur=first;
        NODE prev=NULL;
        if(pos==1)
        {
                temp->link=first;
                return temp;
        }
        int count=0;
        while(count<=pos)
        {
                count=count+1;
                if(count==pos)
                {
                        temp->link=cur;
                        prev->link=temp;
                        return first;
                }
                cur=cur->link;
                if(count==1)prev=first;
                if(count>1)prev=prev->link;
        }
        printf("INVALID POSITION!\n");
        free(temp);
        return first;
}
void display(NODE first)
{
        NODE cur=first;
        printf("The list is:\n----------\n");
        while(cur!=NULL)
        {
                printf("%d\n",cur->info);
                cur=cur->link;
        }
}

void main()
{
        int choice,c=1,item,pos;
        NODE first=NULL;
        while(c==1)
        {
```

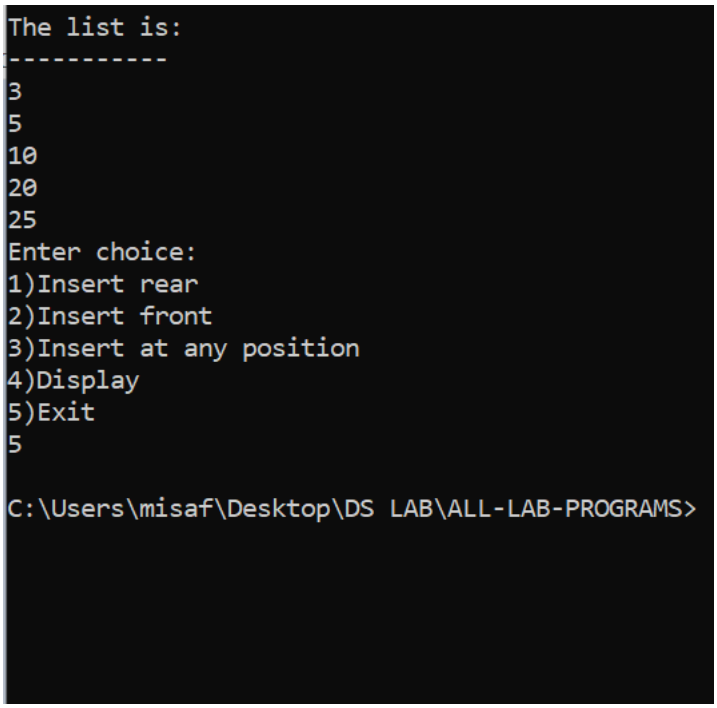
```
            printf("Enter choice:\n1)Insert rear\n2)Insert front\n3)Insert at any
position\n4)Display\n5)Exit\n");
            scanf("%d",&choice);
            switch(choice)
            {
                  case 1:
                        printf("Enter item :\n");
                        scanf("%d",&item);
                        first=insert_rear(first,item);
                        break;
                  case 2:
                        printf("Enter item:\n");
                        scanf("%d",&item);
                        first=insert_front(first,item);
                        break;
                  case 3:
                        printf("Enter position\n");
                        scanf("%d",&pos);
                        printf("Enter item:\n");
                        scanf("%d",&item);
                        first=insert_pos(first,item,pos);
                        break;
                  case 4:
                        display(first);
                        break;
                  case 5:
                        c=0;
                        break;
                  default:printf("Invalid choice!\n");
            }
      }
}
```

OUTPUT:

```
C:\Users\misaf\Desktop\DS LAB\ALL-LAB-PROGRAMS>lab5c
Enter choice:
1)Insert rear
2)Insert front
3)Insert at any position
4)Display
5)Exit
1
Enter item :
10
Enter choice:
1)Insert rear
2)Insert front
3)Insert at any position
4)Display
5)Exit
1
Enter item :
20
Enter choice:
1)Insert rear
2)Insert front
3)Insert at any position
4)Display
5)Exit
2
Enter item:
5
```

```
Enter choice:
1)Insert rear
2)Insert front
3)Insert at any position
4)Display
5)Exit
4
The list is:
-----------
5
10
20
Enter choice:
1)Insert rear
2)Insert front
3)Insert at any position
4)Display
5)Exit
3
Enter position
4
Enter item:
25
Enter choice:
1)Insert rear
2)Insert front
3)Insert at any position
4)Display
5)Exit
4
```

```
The list is:
-----------
5
10
20
25
Enter choice:
1)Insert rear
2)Insert front
3)Insert at any position
4)Display
5)Exit
3
Enter position
1
Enter item:
3
Enter choice:
1)Insert rear
2)Insert front
3)Insert at any position
4)Display
5)Exit
4
```



---

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.

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("MEMORY FULL!\n");
                exit(0);
        }
        return x;
}
NODE insert_rear(NODE first,int item)
{
        NODE temp=getnode();
        temp->info=item;
        temp->link=NULL;
        NODE cur=first;
        if(first==NULL)return temp;
        while(cur->link!=NULL)
        {
                cur=cur->link;
        }
        cur->link=temp;
        return first;

}
NODE insert_front(NODE first,int item)
{
        NODE temp=getnode();
        temp->info=item;
        if(first==NULL)return temp;
        temp->link=first;
        return temp;
}
NODE insert_pos(NODE first,int item,int pos)
{
        NODE temp=getnode();
        temp->info=item;
        temp->link=NULL;
        NODE cur=first;
        NODE prev=NULL;
        if(pos==1)
        {
                temp->link=first;
                return temp;
```

```
        }
        int count=0;
        while(count<=pos)
        {
                count=count+1;
                if(count==pos)
                {
                        temp->link=cur;
                        prev->link=temp;
                        return first;
                }
                cur=cur->link;
                if(count==1)prev=first;
                if(count>1)prev=prev->link;
        }
        printf("INVALID POSITION!\n");
        free(temp);
        return first;
}
NODE delete_first(NODE first)
{
        if(first==NULL)
        {
                printf("LIST EMPTY!\n");
                return first;
        }
        NODE cur=first;
        printf("Item deleted: %d\n",first->info);
        cur=cur->link;
        free(first);
        return cur;
}
NODE delete_last(NODE first)
{
        if(first==NULL)
        {
                printf("LIST EMPTY!\n");
                return first;
        }
        NODE cur=first;
        NODE prev=NULL;
        int c=0;
        while(cur->link!=NULL)
```

```
        {

                cur=cur->link;
                c++;
                if(c==1)prev=first;
                else
                {
                        prev=prev->link;
                }

        }
        printf("Item deleted: %d\n",cur->info);
        free(cur);
        prev->link=NULL;
        return first;

}
NODE delete_pos(NODE first,int pos)
{
        NODE cur=first;
        NODE prev=first;
        int count=1;
        if(pos==1)
        {
                cur=cur->link;
                printf("Deleted item: %d\n",first->info);
                free(first);
                return cur;
        }
        cur=cur->link;
        while(cur!=NULL)
        {
                count++;
                if(pos==count)
                {
                        prev->link=cur->link;
                        printf("Deleted item: %d\n",cur->info);
                        free(cur);
                        return first;
                }
                cur=cur->link;
                prev=prev->link;
        }
```

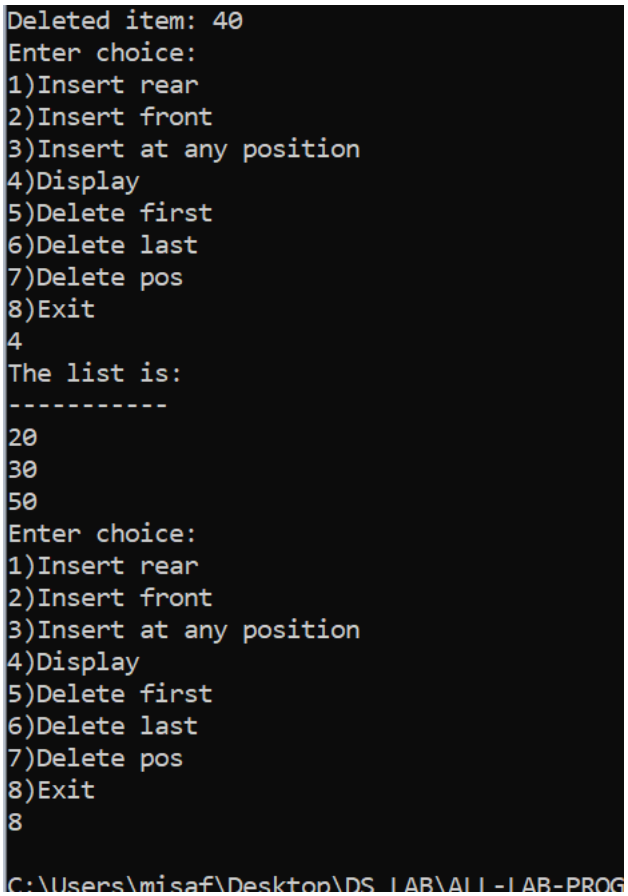
```
        printf("Position not found!\n");
        return first;

}
void display(NODE first)
{
        NODE cur=first;
        printf("The list is:\n----------\n");
        while(cur!=NULL)
        {
                printf("%d\n",cur->info);
                cur=cur->link;
        }
}

void main()
{
        int choice,c=1,item,pos;
        NODE first=NULL;
        while(c==1)
        {
                printf("Enter choice:\n1)Insert rear\n2)Insert front\n3)Insert at any
position\n4)Display\n5)Delete first\n6)Delete last\n7)Delete pos\n8)Exit\n");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1:
                                printf("Enter item :\n");
                                scanf("%d",&item);
                                first=insert_rear(first,item);
                                break;
                        case 2:
                                printf("Enter item:\n");
                                scanf("%d",&item);
                                first=insert_front(first,item);
                                break;
                        case 3:
                                printf("Enter position\n");
                                scanf("%d",&pos);
                                printf("Enter item:\n");
                                scanf("%d",&item);
                                first=insert_pos(first,item,pos);
                                break;
```

```
                case 4:
                        display(first);
                        break;
                case 5:
                        first=delete_first(first);
                        break;
                case 6:
                        first=delete_last(first);
                        break;
                case 7:
                        printf("Enter position:\n");
                        scanf("%d",&pos);
                        first=delete_pos(first,pos);
                        break;
                case 8:
                        c=0;
                        break;
                default:printf("Invalid choice!\n");
            }
        }
}
```

OUTPUT:

```
The list is:
-----------
10
20
30
40
50
60
Enter choice:
1)Insert rear
2)Insert front
3)Insert at any position
4)Display
5)Delete first
6)Delete last
7)Delete pos
8)Exit
5
Item deleted: 10
Enter choice:
1)Insert rear
2)Insert front
3)Insert at any position
4)Display
5)Delete first
6)Delete last
7)Delete pos
8)Exit
6
Item deleted: 60
```

```
Enter choice:
1)Insert rear
2)Insert front
3)Insert at any position
4)Display
5)Delete first
6)Delete last
7)Delete pos
8)Exit
4
The list is:
----------
20
30
40
50
Enter choice:
1)Insert rear
2)Insert front
3)Insert at any position
4)Display
5)Delete first
6)Delete last
7)Delete pos
8)Exit
7
Enter position:
3
Deleted item: 40
Enter choice:
```



---

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

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("MEMORY FULL!\n");
		exit(0);
	}
	return x;
}

NODE insert_rear(NODE first, int item)
{
	NODE temp;
	temp = getnode();
	temp->info = item;
	temp->link = NULL;
	if (first == NULL)
		return temp;
	NODE cur = first;
	while (cur->link != NULL)
	{
		cur = cur->link;
	}
	cur->link = temp;
	return first;
}
NODE sort(NODE first)
{
	int n = 0, i, j;
	NODE cur = first;
	if (first == NULL)
	{
		printf("Empty list!\n");
		return first;
	}
	while (cur != NULL)
	{
		cur = cur->link;
		n++;
	}
```

```
        NODE next = NULL;
        int t;
        for (i = 0; i < n - 1; i++)
        {
                cur = first;
                next = cur->link;
                for (j = 0; j < n - 1 - i; j++)
                {
                        if (cur->info > next->info)
                        {
                                t = cur->info;
                                cur->info = next->info;
                                next->info = t;
                        }
                        next = next->link;
                        cur = cur->link;
                }
        }
        return first;
}
NODE reverse(NODE first)
{
        NODE cur = first;
        NODE next = NULL, prev = NULL;
        while (cur != NULL)
        {
                next = cur->link;
                cur->link = prev;
                prev = cur;
                cur = next;
        }
        return prev;
}
NODE concat(NODE first1, NODE first2)
{
        NODE cur = first1;
        if (first1 == NULL)
        {
                return first2;
        }
        while (cur->link != NULL)
        {
                cur = cur->link;
```

```
        }
        cur->link = first2;
        return first1;
}
void display(NODE first)
{
        NODE cur = first;
        while (cur != NULL)
        {
                printf("%d\n", cur->info);
                cur = cur->link;
        }
}
void main()
{
        int choice, c = 1, item;
        NODE first1 = NULL, first2 = NULL;
        while (c == 1)
        {
                printf("Enter your choice:\n");
                printf("1)Insert in list1\n2)Insert in list2\n3)Sort list1\n4)Sort list2\n5)Reverse
list1\n6)Reverse list2\n7)Concatenate\n8)Display list1\n9)Display list2\n10)Exit\n");
                scanf("%d", &choice);
                switch (choice)
                {
                case 1:
                        printf("Enter item:\n");
                        scanf("%d", &item);
                        first1 = insert_rear(first1, item);
                        break;
                case 2:
                        printf("Enter item:\n");
                        scanf("%d", &item);
                        first2 = insert_rear(first2, item);
                        break;
                case 3:
                        first1 = sort(first1);
                        printf("After sorting:\n");
                        display(first1);
                        break;
                case 4:
                        first2 = sort(first2);
                        printf("After sorting:\n");
```

```
                display(first2);
                break;
        case 5:
                first1 = reverse(first1);
                printf("After reversing:\n");
                display(first1);
                break;
        case 6:
                first2 = reverse(first2);
                printf("After reversing:\n");
                display(first2);
                break;
        case 7:
                first1 = concat(first1, first2);
                printf("After concatenating:\n");
                display(first1);
                break;
        case 8:
                printf("LIST1:\n-------\n");
                display(first1);
                break;
        case 9:
                printf("LIST2:\n-------\n");
                display(first2);
                break;
        case 10:
                c = 0;
                break;
        default:
                printf("Invalid choice!\n");
        }
    }
}
```

OUTPUT:

```
LIST1:
-------
10
4
20
76
54
100
Enter your choice:
1)Insert in list1
2)Insert in list2
3)Sort list1
4)Sort list2
5)Reverse list1
6)Reverse list2
7)Concatenate
8)Display list1
9)Display list2
10)Exit
3
After sorting:
4
10
20
54
76
100
```

```
Enter your choice:
1)Insert in list1
2)Insert in list2
3)Sort list1
4)Sort list2
5)Reverse list1
6)Reverse list2
7)Concatenate
8)Display list1
9)Display list2
10)Exit
5
After reversing:
100
76
54
20
10
4
```

```
Enter your choice:
1)Insert in list1
2)Insert in list2
3)Sort list1
4)Sort list2
5)Reverse list1
6)Reverse list2
7)Concatenate
8)Display list1
9)Display list2
10)Exit
2
Enter item:
1000
Enter your choice:
1)Insert in list1
2)Insert in list2
3)Sort list1
4)Sort list2
5)Reverse list1
6)Reverse list2
7)Concatenate
8)Display list1
9)Display list2
10)Exit
2
Enter item:
3000
```

```
Enter your choice:
1)Insert in list1
2)Insert in list2
3)Sort list1
4)Sort list2
5)Reverse list1
6)Reverse list2
7)Concatenate
8)Display list1
9)Display list2
10)Exit
7
After concatenating:
100
76
54
20
10
4
1000
3000
Enter your choice:
```

---

8)WAP to implement Stack & Queues using Linked Representation

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("MEMORY FULL!\n");
		exit(0);
	}
	return x;
}
NODE insert_rear(NODE first,int item)
{
	NODE temp=getnode();
	temp->info=item;
	temp->link=NULL;
	NODE cur=first;
	if(first==NULL)return temp;
	while(cur->link!=NULL)
	{
```

```
            cur=cur->link;
        }
        cur->link=temp;
        return first;

}
NODE delete_first(NODE first)
{
        if(first==NULL)
        {
                printf("EMPTY!\n");
                return first;
        }
        NODE cur=first;
        printf("Item deleted: %d\n",first->info);
        cur=cur->link;
        free(first);
        return cur;
}
NODE delete_last(NODE first)
{
        if(first==NULL)
        {
                printf("EMPTY!\n");
                return first;
        }
        NODE cur=first;
        NODE prev=NULL;
        int c=0;
        while(cur->link!=NULL)
        {

                cur=cur->link;
                c++;
                if(c==1)prev=first;
                else
                {
                        prev=prev->link;
                }

        }
        printf("Popped: %d\n",cur->info);
        free(cur);
```

```
        prev->link=NULL;
        return first;

}
void display(NODE first)
{
        NODE cur=first;
        while(cur!=NULL)
        {
                printf("%d\n",cur->info);
                cur=cur->link;
        }
}
void main()
{
        int c=1,choice,item;
        NODE qf=NULL,sf=NULL;
        while(c==1)
        {
                printf("Enter choice:\n");
                printf("1)PUSH into stack\n2)POP out of stack\n3)Display stack\n4)Insert into
queue\n5)Delete from queue\n6)Display Queue\n7)Exit\n");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1:
                                printf("Enter item:\n");
                                scanf("%d",&item);
                                sf=insert_rear(sf,item);
                                break;
                        case 2:
                                sf=delete_last(sf);
                                break;
                        case 3:
                                printf("The STACK is :\n----------\n");
                                display(sf);
                                break;
                        case 4:
                                printf("Enter item:\n");
                                scanf("%d",&item);
                                qf=insert_rear(qf,item);
                                break;
                        case 5:
```

```
                qf=delete_first(qf);
                break;
            case 6:
                printf("The QUEUE is :\n----------\n");
                display(qf);
                break;
            case 7:
                c=0;
                break;
            default:
                printf("Invalid choice!\n");

        }
    }
}
```

OUTPUT:

```
C:\Users\misaf\Desktop\DS LAB\ALL-LAB-PROGRAMS>lab8c
Enter choice:
1)PUSH into stack
2)POP out of stack
3)Display stack
4)Insert into queue
5)Delete from queue
6)Display Queue
7)Exit
2
EMPTY!
Enter choice:
1)PUSH into stack
2)POP out of stack
3)Display stack
4)Insert into queue
5)Delete from queue
6)Display Queue
7)Exit
1
Enter item:
10
Enter choice:
1)PUSH into stack
2)POP out of stack
3)Display stack
4)Insert into queue
5)Delete from queue
6)Display Queue
7)Exit
```

```
7)Exit
1
Enter item:
20
Enter choice:
1)PUSH into stack
2)POP out of stack
3)Display stack
4)Insert into queue
5)Delete from queue
6)Display Queue
7)Exit
1
Enter item:
30
Enter choice:
1)PUSH into stack
2)POP out of stack
3)Display stack
4)Insert into queue
5)Delete from queue
6)Display Queue
7)Exit
3
The STACK is :
-----------
10
20
30
Enter choice:
```

```
1)PUSH into stack
2)POP out of stack
3)Display stack
4)Insert into queue
5)Delete from queue
6)Display Queue
7)Exit
2
Popped: 30
Enter choice:
1)PUSH into stack
2)POP out of stack
3)Display stack
4)Insert into queue
5)Delete from queue
6)Display Queue
7)Exit
5
EMPTY!
Enter choice:
1)PUSH into stack
2)POP out of stack
3)Display stack
4)Insert into queue
5)Delete from queue
6)Display Queue
7)Exit
4
Enter item:
1000
```

```
Enter choice:
1)PUSH into stack
2)POP out of stack
3)Display stack
4)Insert into queue
5)Delete from queue
6)Display Queue
7)Exit
4
Enter item:
2000
Enter choice:
1)PUSH into stack
2)POP out of stack
3)Display stack
4)Insert into queue
5)Delete from queue
6)Display Queue
7)Exit
4
Enter item:
3000
Enter choice:
1)PUSH into stack
2)POP out of stack
3)Display stack
4)Insert into queue
5)Delete from queue
6)Display Queue
7)Exit
```

```
6)Display Queue
7)Exit
6
The QUEUE is :
-----------
1000
2000
3000
Enter choice:
1)PUSH into stack
2)POP out of stack
3)Display stack
4)Insert into queue
5)Delete from queue
6)Display Queue
7)Exit
5
Item deleted: 1000
```

---

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.

c) Delete the node based on a specific value. d) Display the contents of the list

CODE:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
	int info;
	struct node *llink;
	struct node *rlink;
};
typedef struct node *NODE;

NODE getnode()
{
	NODE x=(NODE)malloc(sizeof(struct node));
	if(x==NULL)
	{
		printf("MEMORY FULL!");
		exit(0);
	}
	return x;
}
NODE insert_front(NODE first,int item)
{
	NODE temp=getnode();
	temp->info=item;
	temp->rlink=NULL;
	temp->llink=NULL;
	if(first==NULL)return temp;
	temp->rlink=first;
	return temp;
}
NODE insert_rear(NODE first,int item)
```

```
{
	NODE temp=getnode();
	temp->info=item;
	temp->rlink=NULL;
	temp->llink=NULL;
	NODE cur=first;
	if(first==NULL)return temp;
	while(cur!=NULL)
	{
		if(cur->rlink==NULL)
		{
			cur->rlink=temp;
			temp->llink=cur;
			break;
		}
		cur=cur->rlink;
	}
	return first;
}
NODE insert_leftof(NODE first,int item,int key)
{
	NODE temp=getnode();
	temp->info=item;
	temp->rlink=NULL;
	temp->llink=NULL;
	if(first==NULL)
	{
		printf("Key doesnt exit!\n");
		return first;
	}
	if(first->info==key)
	{
		temp->rlink=first;
		first->llink=temp;
		return temp;
	}
	NODE cur=first->rlink;
	NODE prev=first;
	while(cur!=NULL)
	{
		if(cur->info==key)
		{
			prev->rlink=temp;
```

```
                temp->llink=prev;
                temp->rlink=cur;
                cur->llink=temp;
                return first;
            }
            cur=cur->rlink;
            prev=prev->rlink;
        }
        printf("Key doesnt exit!\n");
        return first;
}
void display(NODE first)
{
        NODE cur=first;
        while(cur!=NULL)
        {
            printf("%d\n",cur->info);
            cur=cur->rlink;
        }
}

NODE delete_key(NODE first,int key)
{
        int count=0;
        NODE cur=first;
        NODE prev=NULL,next=NULL;
        if(first==NULL)
        {
            printf("Empty!\n");
            return first;
        }
        while(first->info==key)
        if(first->info==key)
        {
            cur=first;
            first=first->rlink;
            count++;
            free(cur);
        }

        cur=first;
        while(cur!=NULL)
        {
```

```
            if(cur->info==key)
            {
                  count++;
                  prev=cur->llink;
                  next=cur->rlink;
                  prev->rlink=next;
                  next->llink=prev;
                  free(cur);
                  cur=next;
            }
            else
            {
                  cur=cur->rlink;
            }

      }

      printf("%d deleted from %d places\n",key,count);
      return first;

}
void main()
{
      int choice,c=1,item,key;
      NODE first=NULL;
      while(c==1)
      {
            printf("Enter choice:\n1)Insert rear\n2)Insert front\n3)Insert left of
key\n4)Display\n5)Delete all key\n6)Exit\n");
            scanf("%d",&choice);
            switch(choice)
            {
                  case 1:
                        printf("Enter item:\n");
                        scanf("%d",&item);
                        first=insert_rear(first,item);
                        break;
                  case 2:
                        printf("Enter item:\n");
                        scanf("%d",&item);
                        first=insert_front(first,item);
                        break;
                  case 3:
```

```
                printf("Enter key :\n");
                scanf("%d",&key);
                printf("Enter item to be inserted left of key:\n");
                scanf("%d",&item);
                first=insert_leftof(first,item,key);
                break;
            case 4:
                display(first);
                break;
            case 5:
                printf("Enter key value to be deleted:\n");
                scanf("%d",&key);
                first=delete_key(first,key);
                break;
            case 6:c=0;break;
            default:printf("Invalid choice!\n");
        }
    }
}
```

OUTPUT:

```
C:\Users\misaf\Desktop\DS\practise>dllc
Enter choice:
1)Insert rear
2)Insert front
3)Insert left of key
4)Display
5)Delete all key
6)Exit
1
Enter item:
10
Enter choice:
1)Insert rear
2)Insert front
3)Insert left of key
4)Display
5)Delete all key
6)Exit
1
Enter item:
20
Enter choice:
1)Insert rear
2)Insert front
3)Insert left of key
4)Display
5)Delete all key
6)Exit
2
Enter item:
```

```
Enter item:
5
Enter choice:
1)Insert rear
2)Insert front
3)Insert left of key
4)Display
5)Delete all key
6)Exit
4
5
10
20
Enter choice:
1)Insert rear
2)Insert front
3)Insert left of key
4)Display
5)Delete all key
6)Exit
3
Enter key :
10
Enter item to be inserted left of key:
6
Enter choice:
1)Insert rear
2)Insert front
3)Insert left of key
4)Display
```

```
5)Delete all key
6)Exit
4
5
6
10
20
Enter choice:
1)Insert rear
2)Insert front
3)Insert left of key
4)Display
5)Delete all key
6)Exit
5
Enter key value to be deleted:
10
10 deleted from 1 places
Enter choice:
1)Insert rear
2)Insert front
3)Insert left of key
4)Display
5)Delete all key
6)Exit
6
```

---

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.

CODE:

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

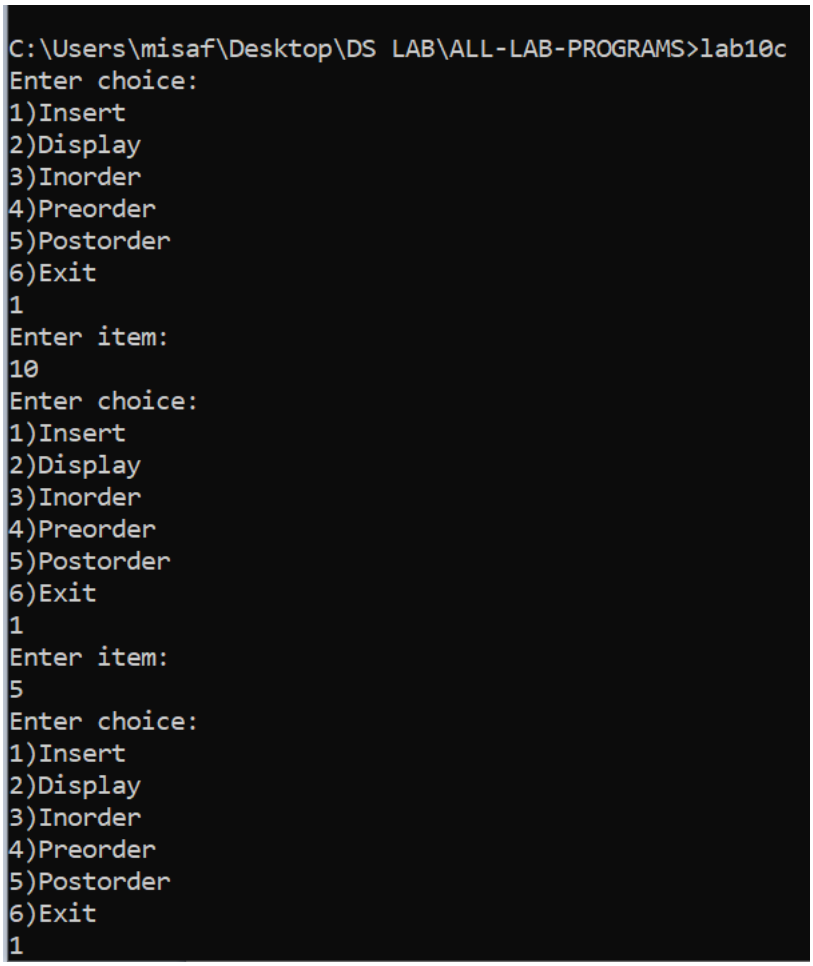
```
struct node
{
	int info;
	struct node *rlink;
	struct node *llink;
};
typedef struct node *NODE;

NODE getnode()
{
	NODE x = (NODE)malloc(sizeof(struct node));
	if (x == NULL)
	{
		printf("MEMORY FULL\n");
		exit(0);
	}
	return x;
}

NODE insert(NODE root, int item)
{
	NODE temp = getnode();
	temp->info = item;
	temp->llink = NULL;
	temp->rlink = NULL;
	if (root == NULL)
	{
		return temp;
	}
	NODE cur = root;
	while (cur != NULL)
	{
		if (item > cur->info)
		{
			if (cur->rlink != NULL)
				cur = cur->rlink;
			else
			{
				cur->rlink = temp;
				break;
			}
		}
		else if (item < cur->info)
```

```
            {
                if (cur->llink != NULL)
                    cur = cur->llink;
                else
                {
                    cur->llink = temp;
                    break;
                }
            }
            else
            {
                printf("Item exists!\n");
                break;
            }
        }
        return root;
}
void inorder(NODE root)
{
        if (root != NULL)
        {
            inorder(root->llink);
            printf("%d\n", root->info);
            inorder(root->rlink);
        }
}
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);
```

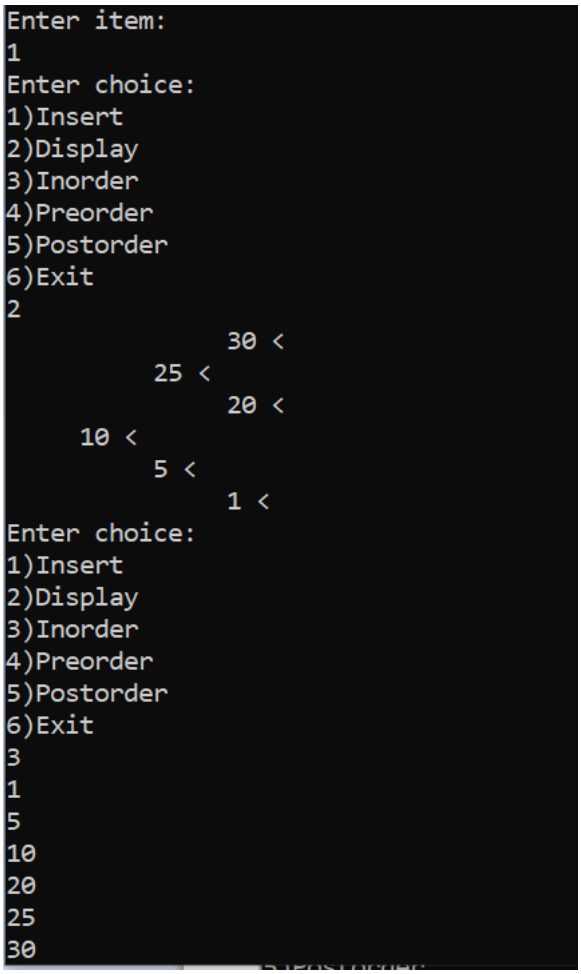
```
            printf("%d\n", root->info);
        }
}
void display(NODE root, int i)
{
        int j;
        if (root != NULL)
        {
            display(root->rlink, i + 1);
            for (j = 0; j < i; j++)
                printf("          ");
            printf("%d <\n", root->info);
            display(root->llink, i + 1);
        }
}
void main()
{
        int choice, c = 1, item;
        NODE root = NULL;
        while (c == 1)
        {
            printf("Enter
choice:\n1)Insert\n2)Display\n3)Inorder\n4)Preorder\n5)Postorder\n6)Exit\n");
            scanf("%d", &choice);
            switch (choice)
            {
            case 1:
                printf("Enter item:\n");
                scanf("%d", &item);
                root = insert(root, item);
                break;
            case 2:
                display(root, 1);
                break;
            case 3:
                inorder(root);
                break;
            case 4:
                preorder(root);
                break;
            case 5:
                postorder(root);
                break;
```

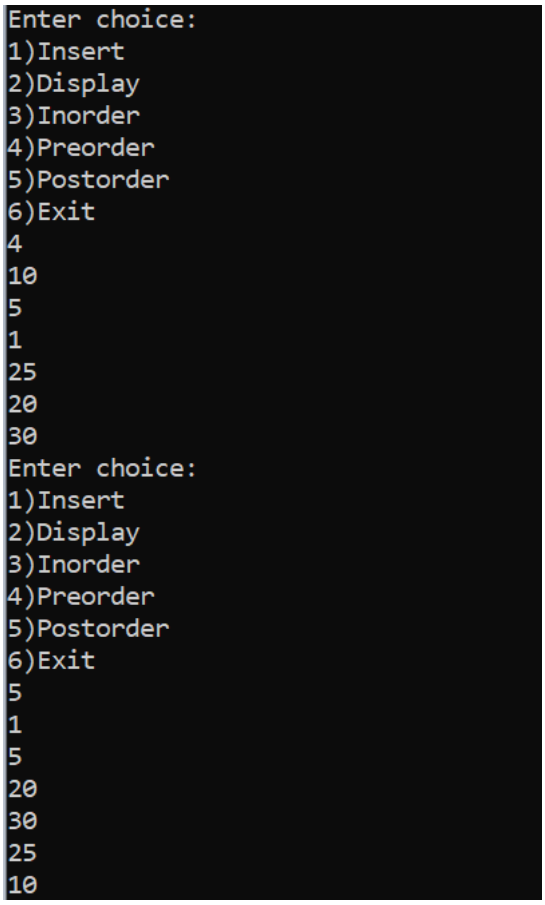
```
            case 6:c=0;break;
            default:
                  printf("Invalid choice!\n");
            }
      }
}
```

OUTPUT:



```
1
Enter item:
25
Enter choice:
1)Insert
2)Display
3)Inorder
4)Preorder
5)Postorder
6)Exit
1
Enter item:
20
Enter choice:
1)Insert
2)Display
3)Inorder
4)Preorder
5)Postorder
6)Exit
1
Enter item:
30
Enter choice:
1)Insert
2)Display
3)Inorder
4)Preorder
5)Postorder
6)Exit
```


Enter item:
1
Enter choice:
1)Insert
2)Display
3)Inorder
4)Preorder
5)Postorder
6)Exit
2
30 <
25 <
20 <
10 <
5 <
1 <
Enter choice:
1)Insert
2)Display
3)Inorder
4)Preorder
5)Postorder
6)Exit
3
1
5
10
20
25
30


Enter choice:
1)Insert
2)Display
3)Inorder
4)Preorder
5)Postorder
6)Exit
4
10
5
1
25
20
30
Enter choice:
1)Insert
2)Display
3)Inorder
4)Preorder
5)Postorder
6)Exit
5
1
5
20
30
25
10