DS-LAB EXPERIMENT 10

NAME: YASH SARANG D6AD/47

Aim: - To write a program to implement stacks ADT using Linked lists.

Theory: -
Instead of using arrays, we can also use linked lists to implement stack. Linked list allocates the memory dynamically. However, time complexity in both the scenarios is same for all the operations i.e., push, pop and peek.
In linked list implementation of stack, the nodes are maintained non-contiguously in the memory. Each node contains a pointer to its immediate successor node in the stack. Stack is said to be overflown if the space left in the memory heap is not enough to create a node.

Operations on Stacks:
1. push (): Insert the element into the linked list of Stack.
2. pop (): Return top element from the Stack and move the top pointer to the second node of linked list or Stack.
3. peek (): Return the top element.
4. display (): Print all elements of Stack.

Algorithm:
1. Insert
- Step 1: Allocate the space for the new node PTR
- Step 2: SET PTR -> DATA = VAL
- Step 3: IF FRONT = NULL SET FRONT = PTR SET FRONT -> NEXT = NULL
- Step 4: END

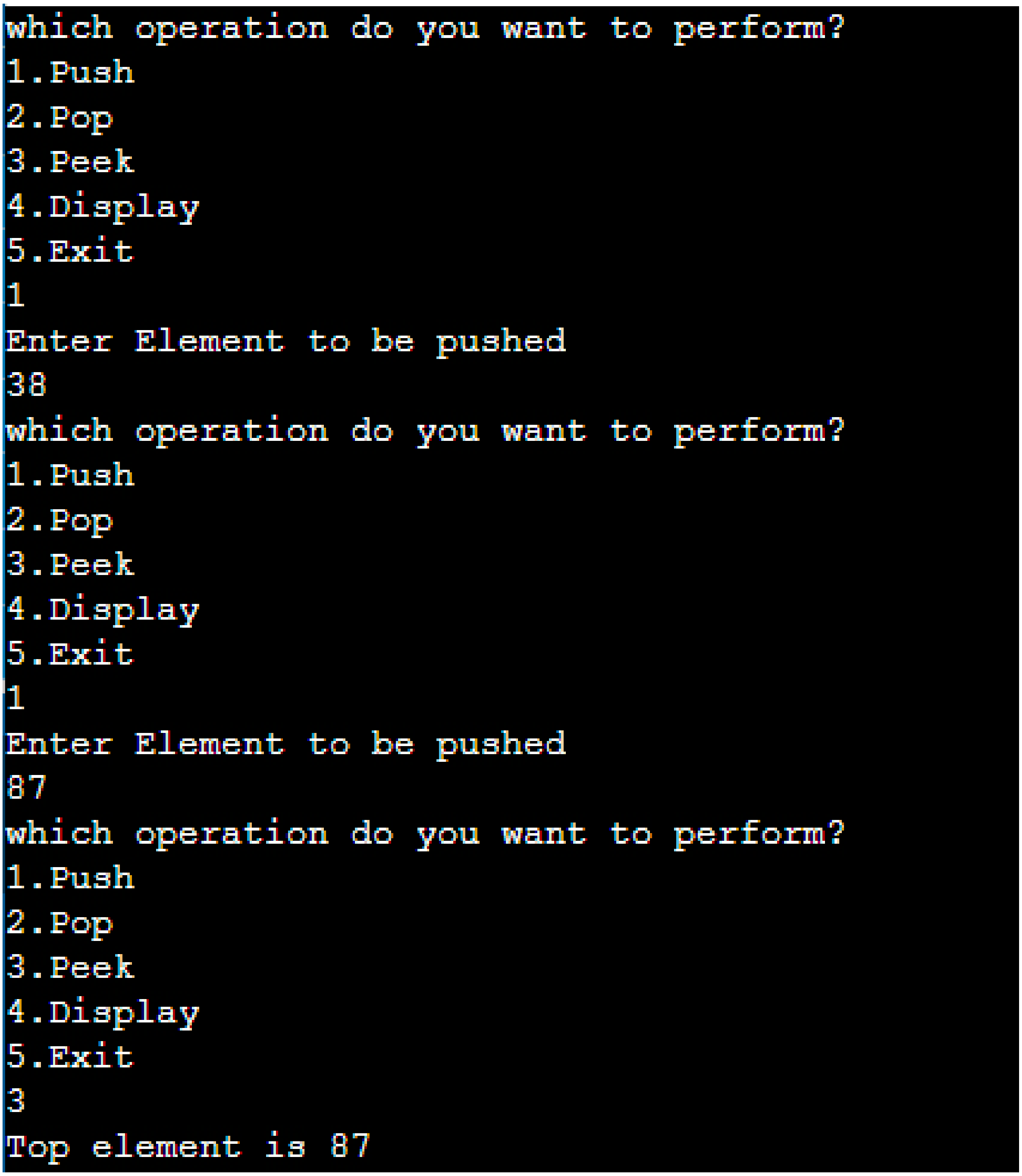
2. Deletion

- Step 1: IF FRONT = NULL Write " Underflow " Go to Step 5 [END OF IF]
- Step 2: SET PTR = FRONT
- Step 3: SET FRONT = FRONT -> NEXT
- Step 4: FREE PTR
- Step 5: END

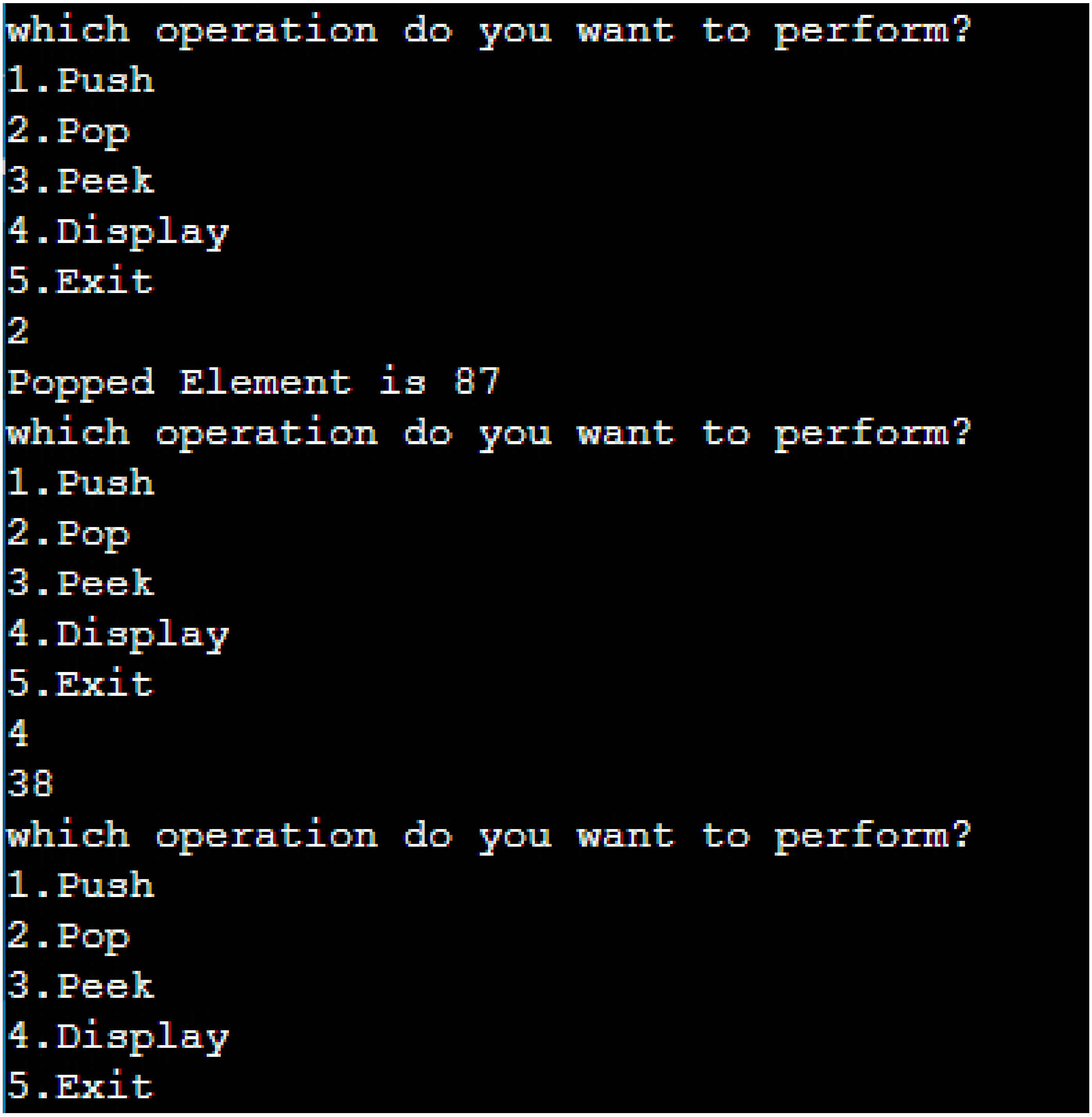
Program:

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

struct Stack
{
    int data;
    struct Stack *next;
};

struct Stack *top = NULL;


// Push Operation
void push()
{
    int x;
    printf("Enter Element to be pushed\n");
    scanf("%d", &x);
    struct Stack *newNode = (struct Stack *)malloc(sizeof(struct Stack));

    newNode->data = x;

    newNode->next = top;

    top = newNode;

}


// Pop Operation

void pop()

{

    struct Stack *temp = (struct Stack *)malloc(sizeof(struct Stack));

    temp = top;

    if (top == NULL)

    {

        printf("Stack is empty\n");

    }

    else

    {

        printf("Popped Element is %d\n", top->data);

        top = top->next;

        free(temp);

    }

}


// Peek Operation

void peek()

{

    if (top == NULL)

    {

        printf("Stack is empty\n");

    }

    else

    {
        printf("Top element is %d\n", top->data);
    }
}

// Display
void display()
{
    struct Stack *temp = (struct Stack *)malloc(sizeof(struct Stack));
    temp = top;
    if (top == NULL)
    {
        printf("Stack is empty\n");
    }
    else
    {
        while (temp != NULL)
        {
            printf("%d ", temp->data);
            temp = temp->next;
        }
    }
    printf("\n");
}

int main()
{
    int opt;
    while (1)
    {

        printf("which operation do you want to perform?\n");
        printf("1.Push\n");
        printf("2.Pop\n");
        printf("3.Peek\n");
        printf("4.Display\n");
        printf("5.Exit\n");
        scanf("%d", &opt);
        switch (opt)
        {
        case 1:
          push();
          break;
        case 2:
          pop();
          break;
        case 3:
          peek();
          break;
        case 4:
          display();
          break;
        case 5:
          exit(0);
          break;
        default:
          printf("Unknown operation\n");
        }
    }
    return 0;
}
```

Output:



```
which operation do you want to perform?
1.Push
2.Pop
3.Peek
4.Display
5.Exit
1
Enter Element to be pushed
38
which operation do you want to perform?
1.Push
2.Pop
3.Peek
4.Display
5.Exit
1
Enter Element to be pushed
87
which operation do you want to perform?
1.Push
2.Pop
3.Peek
4.Display
5.Exit
3
Top element is 87
```



```
which operation do you want to perform?
1.Push
2.Pop
3.Peek
4.Display
5.Exit
2
Popped Element is 87
which operation do you want to perform?
1.Push
2.Pop
3.Peek
4.Display
5.Exit
4
38
which operation do you want to perform?
1.Push
2.Pop
3.Peek
4.Display
5.Exit
```