Lab 1: Implementations of different operations related to the stack.

Theory: A stack is a data structure of ordered items such that items can be inserted and removed only at one end called top of stack (TOS). The items inserted last will be removed first. This mechanism is called Last-In-First-Out (LIFO).

Stack operations:

1. Push: Placing a data item on the top is called "pushing".
   Let, TOP be the Top of Stack, MAX be the size of the stack, item is the item that is to be inserted into the stack.
   A. Start
   B. If TOP=MAX-1 then Display Stack Overflow
   C. Else, TOP=TOP+1
      Stack [TOP] =item.
   D. Stop
2. Pop: Removing an item from the top is called "popping".
   Let TOP be the top of stack.
   A. Start
   B. if TOP=-1 then Display Stack Underflow
   C. Else, Remove the top element from the stack and return this element.
      TOP=TOP-1
   D. Stop
3. Peek: Viewing the item at the top is called "peeking".
   Let TOP be the top of stack.
   A. Start
   B. if TOP=-1 then Display Stack Empty
   C. Else, Return the top element from the stack,
   D. Stop

Source Code:

```
#include <stdio.h>
#define MAX 10
#define true 1
#define false 0
struct Stack
{
   int data[MAX];
   int TOP;
};
void PUSH(struct Stack *S, int ele);
int POP(struct Stack *S);
int PEEK(struct Stack *S);
int isFULL(struct Stack *S);
int isEMPTY(struct Stack *S);
int main()
{
   int choice,ele;
   struct Stack SS;
   SS.TOP = -1;
   do
   {
   printf("\n1.PUSH\n2.POP\n3.PEEK\n4.EXIT\n");
   printf("Enter your choice ");
   scanf("%d",&choice);
   switch(choice)
   {
   case 1:
      printf("Enter Element: ");
      scanf("%d", &ele);
      PUSH(&SS,ele);
      break;

   case 2:
      if (isEMPTY(&SS))
      {
         printf("STACK UNDERFLOW\n");
      }
      else
      {
```

```
            ele = POP(&SS);
            printf("%d was POPPED",ele);
        }
        break;

    case 3:
        if (isEMPTY(&SS))
        {
            printf("STACK EMPTY\n");
        }
        else
        {
            ele = PEEK(&SS);
            printf("%d was at TOP\n",ele);
        }
        break;

    case 4:
        printf("BYE\n");
        break;

    default:
        printf("Enter 1, 2, 3 or 4 only");
        break;
    }
    }while(choice != 4);
    return 0;
}

void PUSH(struct Stack *S, int ele)
{
    if(isFULL(S))
    {
        printf("STACK OVERFLOW\n");
    }
    else
    {
        S->TOP += 1;
        S->data[S->TOP] = ele;
        printf("%d was PUSHED\n",ele);
```

```
    }

}

int POP(struct Stack *S)
{
    int ret = S->data[S->TOP];
    S->TOP -= 1;
    return ret;
}

int PEEK(struct Stack *S)
{
    return S->data[S->TOP];
}

int isFULL(struct Stack *S)
{
    /*
    if (S->TOP == MAX -1)
        return true;
    else
        return false;
    */
    return S->TOP == MAX -1 ? true : false;
}
int isEMPTY(struct Stack *S)
{
    /*
    if(S->TOP == -1)
        return true;
    else
        return false;
    */
    return S->TOP == -1 ? true : false;
}
```

Output:

```
1.PUSH
2.POP
3.PEEK
4.EXIT
Enter your choice 1
Enter Element: 11
11 was PUSHED

1.PUSH
2.POP
3.PEEK
4.EXIT
Enter your choice 2
11 was POPPED
1.PUSH
2.POP
3.PEEK
4.EXIT
Enter your choice 4
BYE
```

Conclusion: Hence, in this lab we implemented stack as an ADT using arrays. We implemented PUSH, POP and PEEK operations successfully.