Experiment:-1.3

Write a program to implement the functions on a stack:

a. PUSH
b. POP
c. OVERFLOW& UNDERFLOW

Student Name:-Neha Sharma **UID:-20BCS4576**
Branch:- CSE-IOT **Section/Group:- A**
Semester:-3rd **Date of Performance:-7/09/2021**
Subject Name:- DS LAB **Subject Code:-20CSP-231**

1. **Aim/Overview of the practical:-** **Write a program to implement the functions on a stack:**
   - PUSH
   - POP
   - OVERFLOW& UNDERFLOW

2. **Task to be done:-** Write a C program to implement stack data structure with push and pop operation. In this post I will explain stack implementation using array in C language.

3. Algorithm/Flowchart:

For PUSH

Step 1: **If Top=Max-1**

Print "Overflow : Stack is full" and Exit

End If

Step 2: **Top=Top+1**

Step 3: **Stack[TOP]=Element**

Step 4: **End**

For POP

Step 1: **If TOP=-1**

Print "Underflow: Stack is empty" and Exit

End if

Step 2: **Set Del_element=Stack[Top]**

Step 3: **Top=Top-1**

Step 4: **Del_Element**

Step 5: **End**

4. Steps for experiment/practical:

```
#include <stdio.h>

#include <stdlib.h>

#include <limits.h>    // For INT_MIN

#define SIZE 100

// Create a stack with capacity of 100 elements

int stack[SIZE];

// Initially stack is empty

int top = -1;

/* Function declaration to perform push and pop on stack */

void push(int element);

int pop();


int main()

{
```

```
int choice, data;

while(1)

{

   /* Menu */

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

   printf("   STACK IMPLEMENTATION PROGRAM   \n");

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

   printf("1. Push\n");

   printf("2. Pop\n");

   printf("3. Size\n");

   printf("4. Exit\n");

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

   printf("Enter your choice: ");


   scanf("%d", &choice);


   switch(choice)

   {

      case 1:

         printf("Enter data to push into stack: ");
```

```
    scanf("%d", &data);

    // Push element to stack
    push(data);
    break;

case 2:
    data = pop();

    // If stack is not empty
    if (data != INT_MIN)
        printf("Data => %d\n", data);
    break;

case 3:
    printf("Stack size: %d\n", top + 1);
    break;
```

```
        case 4:
            printf("Exiting from app.\n");
            exit(0);
            break;

        default:
            printf("Invalid choice, please try again.\n");
    }
    printf("\n\n");
  }
 return 0;
}
 /**
 * Functiont to push a new element in stack.
 */
void push(int element)
{
  // Check stack overflow
  if (top >= SIZE)
```

```
  {
     printf("Stack Overflow, can't add more element element to stack.\n");
     return;
  }
  // Increase element count in stack
  top++;
  // Push element in stack
  stack[top] = element;
  printf("Data pushed to stack.\n");
}
/**
 * Function to pop element from top of stack.
 */
int pop()
{
  // Check stack underflow
  if (top < 0)
  {
     printf("Stack is empty.\n");
```

```
    return INT_MIN;
  }
  return stack[top--];
}
```

5. **Output: Image of sample output to be attached here**

```
-----------------------------------
   STACK IMPLEMENTATION PROGRAM
-----------------------------------
1. Push
2. Pop
3. Size
4. Exit
-----------------------------------
Enter your choice: 1
Enter data to push into stack: 4
Data pushed to stack.


-----------------------------------
   STACK IMPLEMENTATION PROGRAM
-----------------------------------
1. Push
2. Pop
3. Size
4. Exit
-----------------------------------
Enter your choice: 2
Data => 4


-----------------------------------
   STACK IMPLEMENTATION PROGRAM
-----------------------------------
1. Push
2. Pop
3. Size
4. Exit
```

```
4. Exit
-----------------------------------
Enter your choice: 3
Stack size: 0


-----------------------------------
   STACK IMPLEMENTATION PROGRAM
-----------------------------------
1. Push
2. Pop
3. Size
4. Exit
-----------------------------------
Enter your choice: 4
Exiting from app.


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

Learning outcomes (What I have learnt):

Add item to the top of the stack.

- **Remove an item from the top of the stack.**
- **Evaluation of expressions.**
- **Backtracking.**
- **Runtime memory management.**

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			