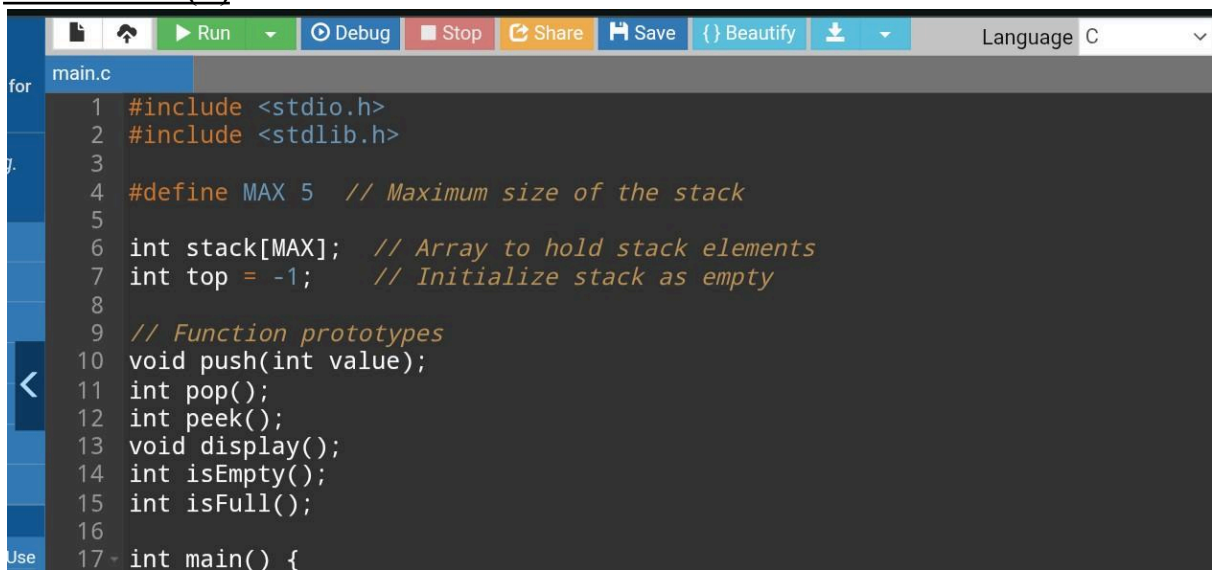
Experiment No: 05
Experiment Name: Stack Operation

Objective:
To understand the concept of Stack Data Structure and to implement its basic operations such as Push, Pop, Peek/Top, and Display using a suitable programming language (C/C++/Java).

Source Code:(C)




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

#define MAX 5  // Maximum size of the stack

int stack[MAX];  // Array to hold stack elements
int top = -1;    // Initialize stack as empty

// Function prototypes
void push(int value);
int pop();
int peek();
void display();
int isEmpty();
int isFull();

int main() {
    int choice, value;

    while (1) {
        printf("\nStack Operations Menu:\n");
        printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Peek\n");
        printf("4. Display\n");
        printf("5. Check if Stack is Empty\n");
        printf("6. Check if Stack is Full\n");
        printf("7. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1: // Push operation
                printf("Enter value to push: ");
                scanf("%d", &value);
                push(value);
                break;

            case 2: // Pop operation
                value = pop();
                if (value != -1)
                    printf("Popped element: %d\n", value);
                break;
```

```
44
45              case 3: // Peek operation
46                  value = peek();
47                  if (value != -1)
48                      printf("Top element is: %d\n", value);
49                  break;
50
51              case 4: // Display operation
52                  display();
53                  break;
54
55              case 5: // Check if stack is empty
56                  if (isEmpty())
57                      printf("Stack is empty.\n");
58                  else
59                      printf("Stack is not empty.\n");
60                  break;
61
```

```
61
62              case 6: // Check if stack is full
63                  if (isFull())
64                      printf("Stack is full.\n");
65                  else
66                      printf("Stack is not full.\n");
67                  break;
68
69              case 7: // Exit
70                  printf("Exiting...\n");
71                  exit(0);
```

```
72
73              default:
74                  printf("Invalid choice! Please try again.\n");
75          }
76      }
77
78      return 0;
79  }
80
81  // Push operation
82  void push(int value) {
83      if (isFull())
84          printf("Stack Overflow! Cannot push %d\n", value);
85      else {
```

```
86          stack[++top] = value;
87          printf("Pushed %d onto the stack\n", value);
88      }
89  }
90
91  // Pop operation
92  int pop() {
93      if (isEmpty()) {
94          printf("Stack Underflow! Cannot pop\n");
95          return -1;
96      } else
```

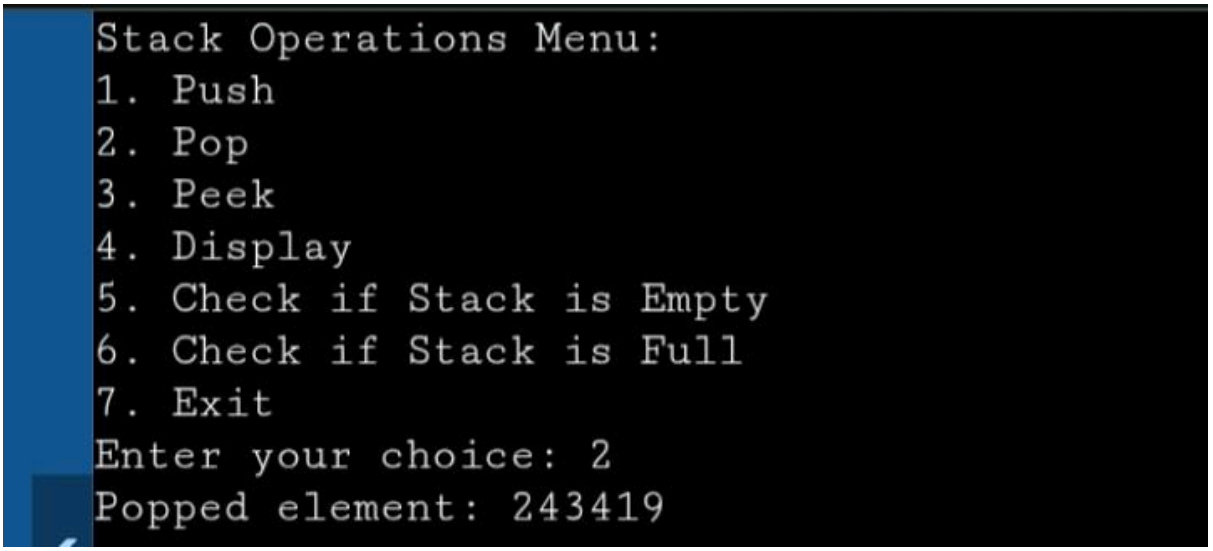
```
97          return stack[top--];
98      }
99
100     // Peek operation
101     int peek() {
102         if (isEmpty()) {
103             printf("Stack is empty! Cannot peek\n");
104             return -1;
105         } else
106             return stack[top];
107     }
108
109     // Display operation
110     void display() {
111         if (isEmpty())
112             printf("Stack is empty! Nothing to display\n");
113         else {
114             printf("Stack elements from top to bottom:\n");
115             for (int i = top; i >= 0; i--)
116                 printf("%d ", stack[i]);
117             printf("\n");
118         }
119     }
120
121     // Check if stack is full
122     int isFull() {
123         return top == MAX - 1;
124     }
125
126     // Check if stack is empty
127     int isEmpty() {
128         return top == -1;
129     }
130
```

Fig 01: Stack Operation

Output:

```
input

Stack Operations Menu:
1. Push
2. Pop
3. Peek
4. Display
5. Check if Stack is Empty
6. Check if Stack is Full
7. Exit
Enter your choice: 1
Enter value to push: 243419
Pushed 243419 onto the stack
```


Stack Operations Menu:
1. Push
2. Pop
3. Peek
4. Display
5. Check if Stack is Empty
6. Check if Stack is Full
7. Exit
Enter your choice: 2
Popped element: 243419

```
Stack Operations Menu:
1. Push
2. Pop
3. Peek
4. Display
5. Check if Stack is Empty
6. Check if Stack is Full
7. Exit
Enter your choice: 3
Stack is empty! Cannot peek
```

```
Stack Operations Menu:
1. Push
2. Pop
3. Peek
4. Display
5. Check if Stack is Empty
6. Check if Stack is Full
7. Exit
Enter your choice: 4
Stack is empty! Nothing to display
```

```
Stack Operations Menu:
1. Push
2. Pop
3. Peek
4. Display
5. Check if Stack is Empty
6. Check if Stack is Full
7. Exit
Enter your choice: 5
Stack is empty.
```

```
Stack Operations Menu:
1. Push
2. Pop
3. Peek
4. Display
5. Check if Stack is Empty
6. Check if Stack is Full
7. Exit
Enter your choice: 6
Stack is not full.
```

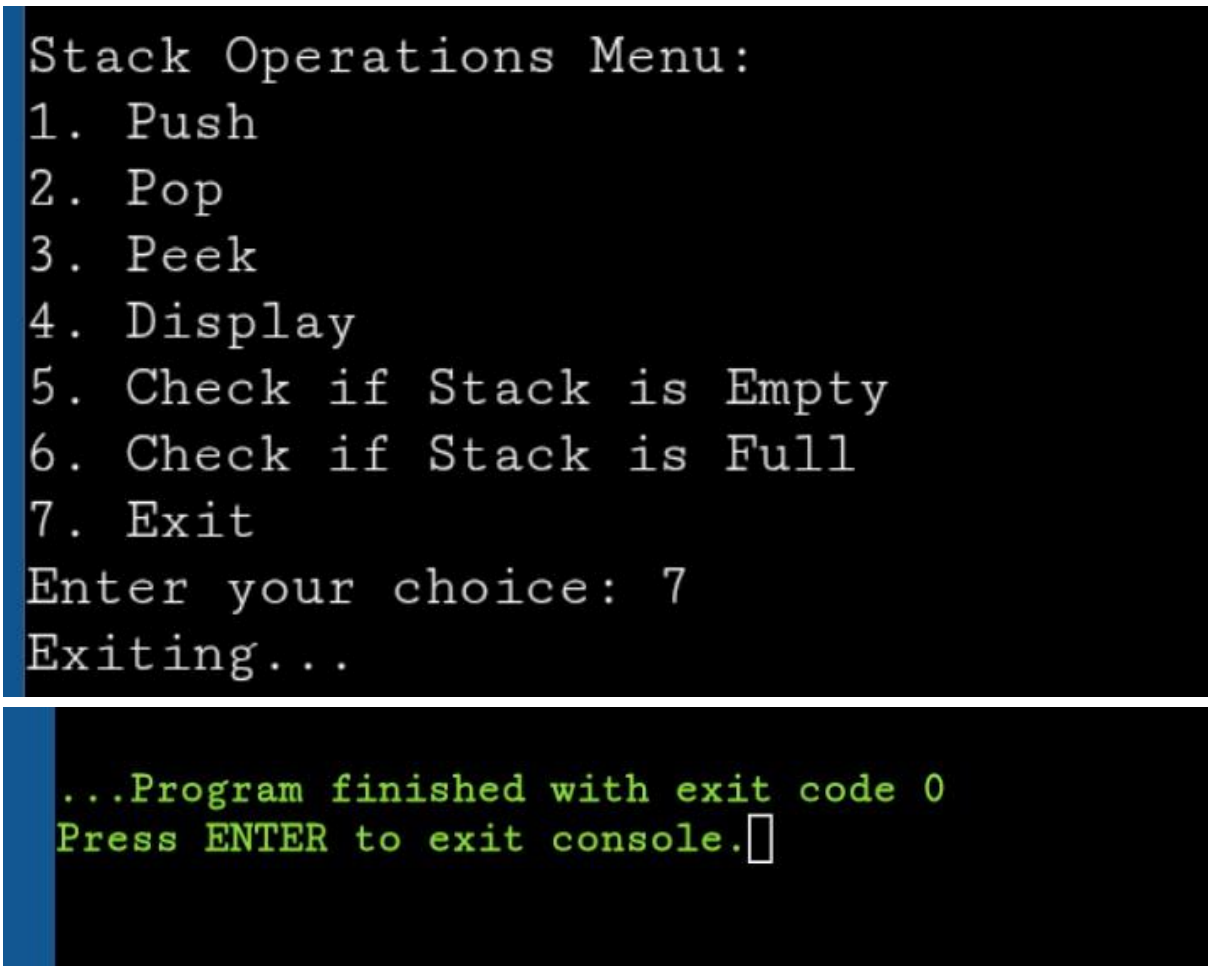


Fig 02: Output

Discussion: In this experiment, the stack data structure was implemented successfully using an array. The stack follows the LIFO (Last In, First Out) principle, meaning the last element pushed onto the stack is the first to be removed. The operations Push, Pop, Peek, and Display were performed and tested to ensure proper functionality. Overflow and underflow conditions were handled correctly, preventing errors when the stack was full or empty. The experiment also demonstrated how the top pointer keeps track of the current position in the stack. This exercise helped in understanding the practical application of stacks in programming, such as in recursion, expression evaluation, and memory management. Overall, it provided a clear insight into the importance and working of stack operations in computer science.