Data Structures : Algorithms and Applications(Lab)

Experiment No: 1

Aim: Implementation of Stack Data Structure using array.

Theory:
Definition:
A stack is a linear data structure that follows the Last In, First Out (LIFO) principle. This means that the last element added to the stack will be the first one to be removed.

Properties:
LIFO (Last In, First Out): The most recently added element is the first to be removed.
Operations: Stacks primarily support the following operations:
Push: Add an element to the top of the stack.
Pop: Remove and return the top element from the stack.
Peek/Top: Return the top element without removing it.
IsEmpty: Check if the stack is empty.
Size: Return the number of elements in the stack.
Common Operations:
Push:Adds an element to the top of the stack.
Pop:Removes the top element from the stack.
Peek:Returns the top element without removing it.
IsEmpty:Checks if the stack is empty.
Size:Returns the number of elements in the stack.

Application:
Stacks are widely used in various applications due to their LIFO nature. Some common applications include:
Infix to Postfix/Prefix conversion.
Evaluation of Postfix/Prefix expressions.

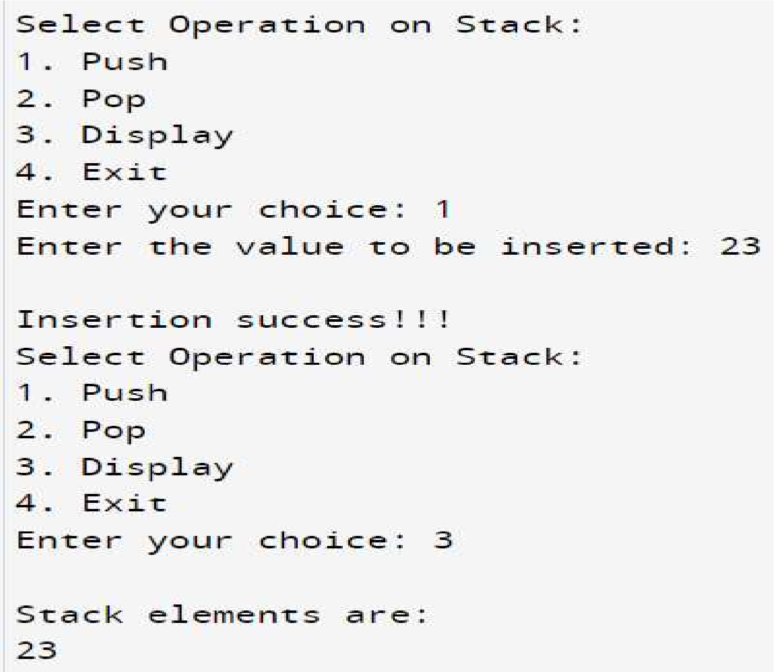
Limitations:

Fixed Size: The array-based implementation has a fixed size, which can be a limitation if the required stack size is unknown or varies significantly.
Memory Inefficiency: If the stack size is overestimated, it can lead to unused memory. Conversely, underestimation can lead to stack overflow.

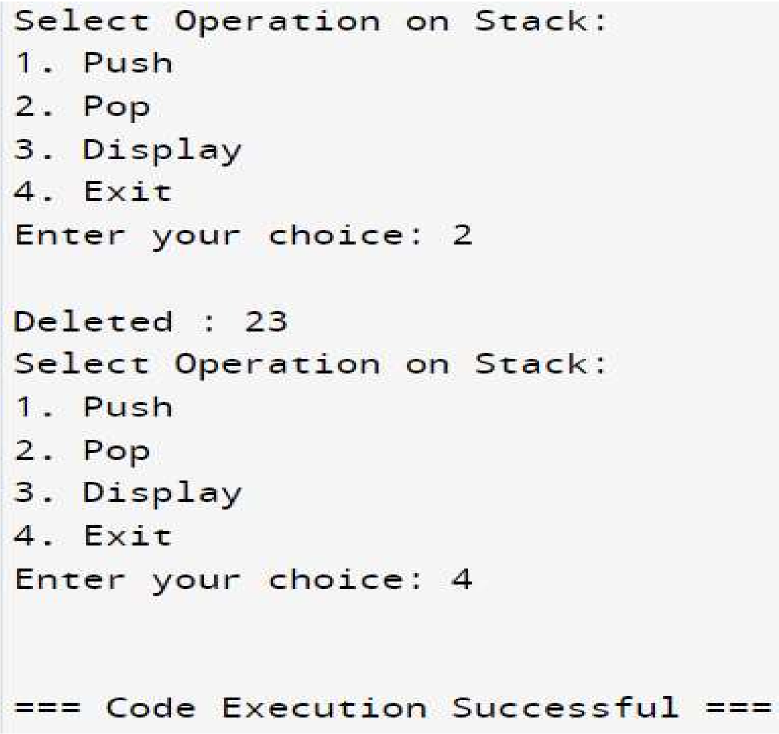
Code:

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

#define SIZE 10

void push(int);
void pop();
void display();

int stack[SIZE], top = -1;

void main() {
    int value, choice;
    clrscr();
    while (1) {
        printf("\n\nSelect Operation on Stack:\n");
        printf("1. Push\n2. Pop\n3. Display\n4. Exit");
        printf("\nEnter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter the value to be inserted: ");
                scanf("%d", &value);
                push(value);
                break;
            case 2:
                pop();
                break;
            case 3:
                display();
                break;
            case 4:
                exit(0);
            default:
                printf("\nWrong selection!!! Try again!!!");
        }
    }
}
void push(int value) {
    if (top == SIZE - 1) {
        printf("\nStack is Full!!! Insertion is not possible!!!");
    } else {
        top++;
        stack[top] = value;
        printf("\nInsertion success!!!");
    }
}
void pop() {
    if (top == -1) {
        printf("\nStack is Empty!!! Deletion is not possible!!!");
    } else {
        printf("\nDeleted : %d", stack[top]);
        top--;
    }
}
void display() {
    if (top == -1) {
        printf("\nStack is Empty!!!");
    } else {
        int i;
        printf("\nStack elements are:\n");
        for (i = top; i >= 0; i--) {
            printf("%d\n", stack[i]);
        }
    }
}
```

Output :



```
Select Operation on Stack:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the value to be inserted: 23

Insertion success!!!
Select Operation on Stack:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3

Stack elements are:
23
```



```
Select Operation on Stack:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2

Deleted : 23
Select Operation on Stack:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 4


=== Code Execution Successful ===
```

Conclusion :

The stack data structure is a fundamental concept in computer science, characterized by its Last In, First Out (LIFO) behavior. This implementation using an array in C provides a clear example of how to manage a stack with basic operations such as push, pop, and display.