**Batch: C3 Roll No.: 16010123217**

**Experiment / assignment / tutorial No. 2**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Title:**  Implementation of Basic operations on stack using Array - Create, Insert, Delete, Peek. |

**Objective:** To implement Basic Operations on Stack i.e. Create, Push, Pop, Peek

**Expected Outcome of Experiment:**

|  |  |
| --- | --- |
| **CO** | **Outcome** |
| 1 | Explain the different data structures used in problem solving |

**Books/ Journals/ Websites referred:**

1. *Fundamentals Of Data Structures In C –* Ellis Horowitz, Satraj Sahni, Susan Anderson-Fred
2. *An Introduction to data structures with applications –* Jean Paul Tremblay,

Paul G. Sorenson

1. *Data Structures A Pseudo Approach with C –* Richard F. Gilberg & Behrouz A. Forouzan
2. [*https://www.cprogramming.com/tutorial/computersciencetheory/stack.html*](https://www.cprogramming.com/tutorial/computersciencetheory/stack.html)
3. [*https://www.geeksforgeeks.org/stack-data-structure-introduction-program/*](https://www.geeksforgeeks.org/stack-data-structure-introduction-program/)
4. [*https://www.thecrazyprogrammer.com/2013/12/c-program-for-array-representation-of-stack-push-pop-display.html*](https://www.thecrazyprogrammer.com/2013/12/c-program-for-array-representation-of-stack-push-pop-display.html)
5. [*https://www.naukri.com/code360/library/application-of-stack*](https://www.naukri.com/code360/library/application-of-stack)
6. [*https://www.enjoyalgorithms.com/blog/application-of-stack-data-structure-in-programming*](https://www.enjoyalgorithms.com/blog/application-of-stack-data-structure-in-programming)

**Abstract**:

A Stack is an ordered collection of elements , but it has a special feature that

deletion and insertion of elements can be done only from one end, called the

top of the stack(TOP). The order may be LIFO(Last In First Out) or FILO(First In Last Out).

Students need to first try and understand the implementation of using arrays. Once comfortable with the concept, they can further implement stacks using linked list as well.

**Related Theory: -**

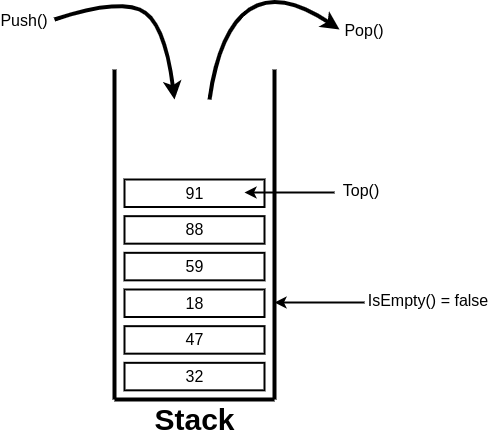
Stack is a linear data structure which follows a particular order in which the operations are performed. It works on the mechanism of Last in First out (LIFO).

**List 5 Real Life Examples where we use stack:**

**Ans:**

1. Back and Forward buttons in a web browser
2. UNDO/REDO functionality in text editors and image editing software
3. Memory management in computer programming
4. Implementing recursion in programming
5. Matching HTML tags in web development

**Diagram:**



**Explain Stack ADT:**

The abstract datatype is special kind of datatype, whose behavior is defined by a set of values and set of operations. The keyword “Abstract” is used as we can use these datatypes, we can perform different operations. But how those operations are working that is totally hidden from the user. The ADT is made of with primitive datatypes, but operation logics are hidden.

A stack is a sequence of items that are accessible at only one end of the sequence. It is an ordered group of homogeneous items or elements. Elements are added to and removed from a specially designated end called the top of the stack (the most recently added items are at the top of the stack). The last element to be added is the first to be removed (LIFO: Last In, First Out).

These are few operations or functions of the Stack ADT.

1. isFull(), This is used to check whether stack is full or not
2. isEmpry(), This is used to check whether stack is empty or not
3. push(x), This is used to push x into the stack
4. pop(), This is used to delete one element from top of the stack
5. peek(), This is used to get the top most element of the stack
6. size(), this function is used to get number of elements present into the stack

**Algorithm for creation, insertion, deletion, displaying an element in stack [static implementation]:**

**1. Algorithm for creation**

Algorithm StackType CreateStack()

{

integer StackTop =-1;

return stack;

}

**2. Algorithm for insertion**

Algorithm StackType PushStack(StackType Stack, ElementType Element){

if NotFull(Stack)= True

stack[++StackTop]= Element

Else “Error Message”

}

**3. Algorithm for deletion**

Algorithm ElementType PopStack(StackType stack)

[This algorithm accepts a stack as input and returns ‘Element’ at the top of ‘stack’. ]

{

if NotEmpty(Stack)= True

return Stack[StackTop--]

else print “Error Message”

}

**4. Algorithm for displaying elements**

Abstract ElementType Peek(StackType stack)

[This algorithm accepts a stack as input and returns ‘Element’ at the top of ‘stack’. ]

{

if NotEmpty(Stack)= True

return Stack[StackTop]

else print “Error Message”

}

**Implementation Details:**

**Built-In Functions/Header Files Used: exit(), printf() and scanf()**

**Program source code:**

#include <stdio.h>

#include <stdlib.h>

#define MAX 10 //initialised array of size 10 to show all operations

typedef struct {

int arr[MAX];

int top;

} Stack;

//creating a Stack

void initialize(Stack \*s) {

(\*s).top = -1;

}

//Check whether full

int isFull(Stack \*s) {

return (\*s).top == MAX - 1;

}

//Check whether empty

int isEmpty(Stack \*s) {

return (\*s).top == -1;

}

//Push the element to the top

void push(Stack \*s, int ele) {

if (isFull(s)) {

printf("Stack Overflow\n");

return;

}

(\*s).arr[++((\*s).top)] = ele;

printf("Pushed %d to the stack\n", ele);

}

//Pop the top element

int pop(Stack \*s) {

if (isEmpty(s)) {

printf("Stack Underflow\n");

return -1;

}

return (\*s).arr[((\*s).top)--];

}

//displaying the top element

int peek(Stack \*s) {

if (isEmpty(s)) {

printf("Stack is empty\n");

return -1;

}

return (\*s).arr[(\*s).top];

}

void delete(Stack \*s) {

if (isEmpty(s)) {

printf("Stack is already empty\n");

return;

}

(\*s).top = -1;

printf("Stack deleted\n");

}

//displaying the stack

void display(Stack \*s) {

if (isEmpty(s)) {

printf("Stack is empty\n");

return;

}

printf("Stack elements: ");

for (int i = 0; i <= (\*s).top; i++) {

printf("%d ", (\*s).arr[i]);

}

printf("\n");

}

int main() {

Stack s;

initialize(&s);

int choice, ele;

//Menu-driven options

while (1) {

printf("\nMenu:\n");

printf("1. Push\n");

printf("2. Pop\n");

printf("3. Peek\n");

printf("4. Delete stack\n");

printf("5. Display stack\n");

printf("6. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

//switch-case for user input operations

switch (choice) {

case 1:

printf("Enter the element to be pushed: ");

scanf("%d", &ele);

push(&s, ele);

break;

case 2:

ele = pop(&s);

if (ele != -1) {

printf("Popped element: %d\n", ele);

}

break;

case 3:

ele = peek(&s);

if (ele != -1) {

printf("Top element: %d\n", ele);

}

break;

case 4:

delete(&s);

break;

case 5:

display(&s);

break;

case 6:

exit(0);

default:

printf("Invalid choice, please try again\n");

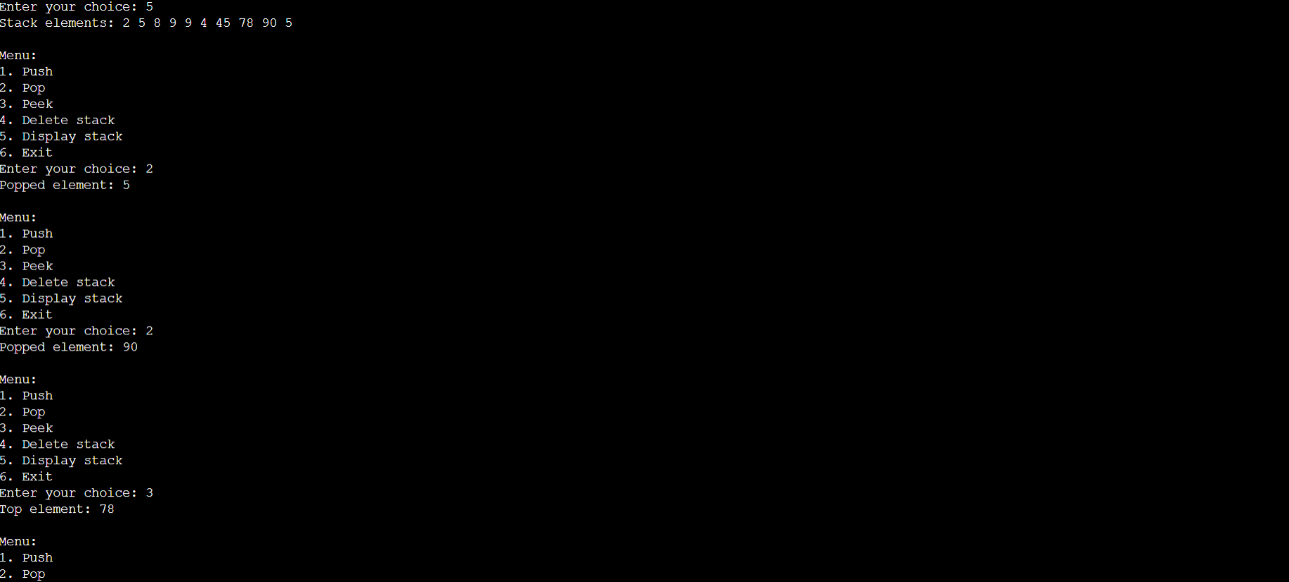
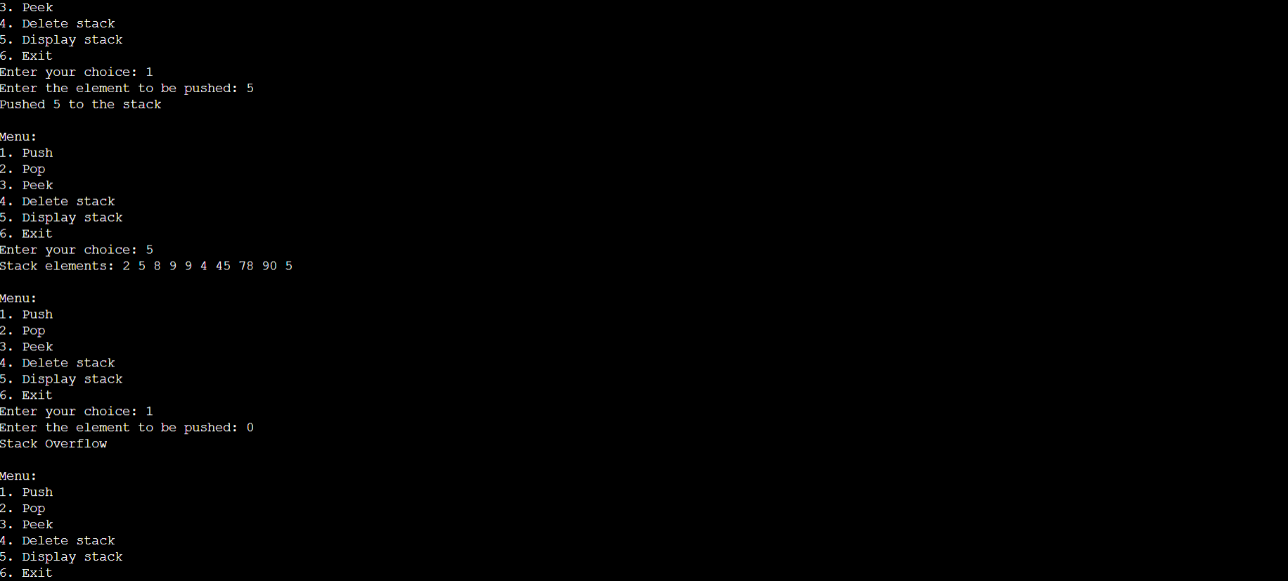
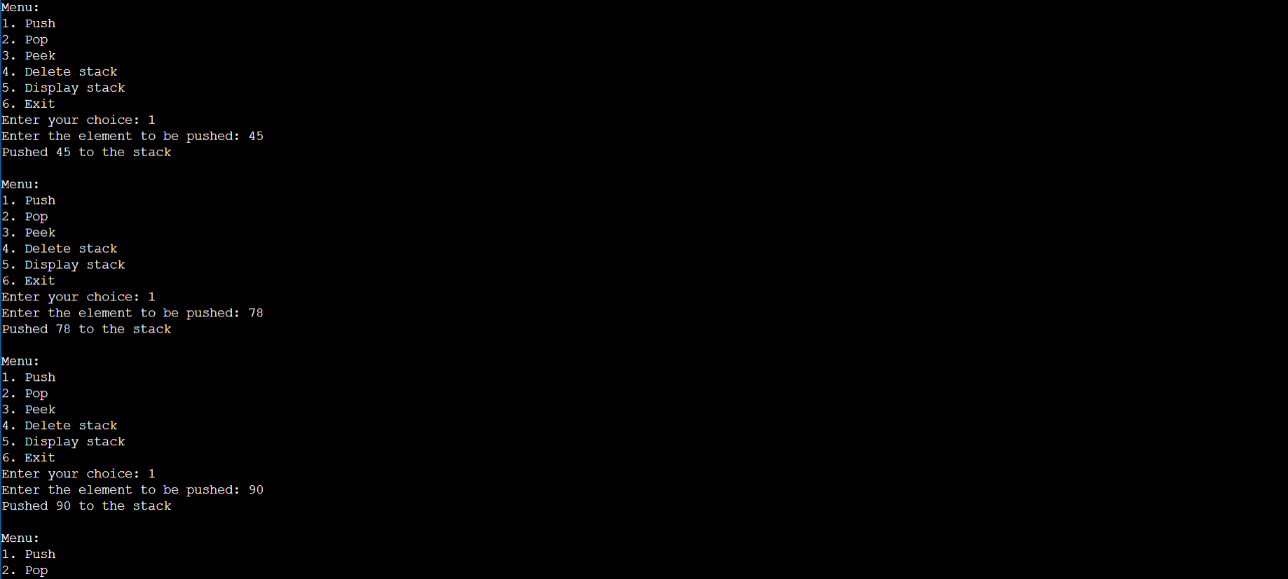
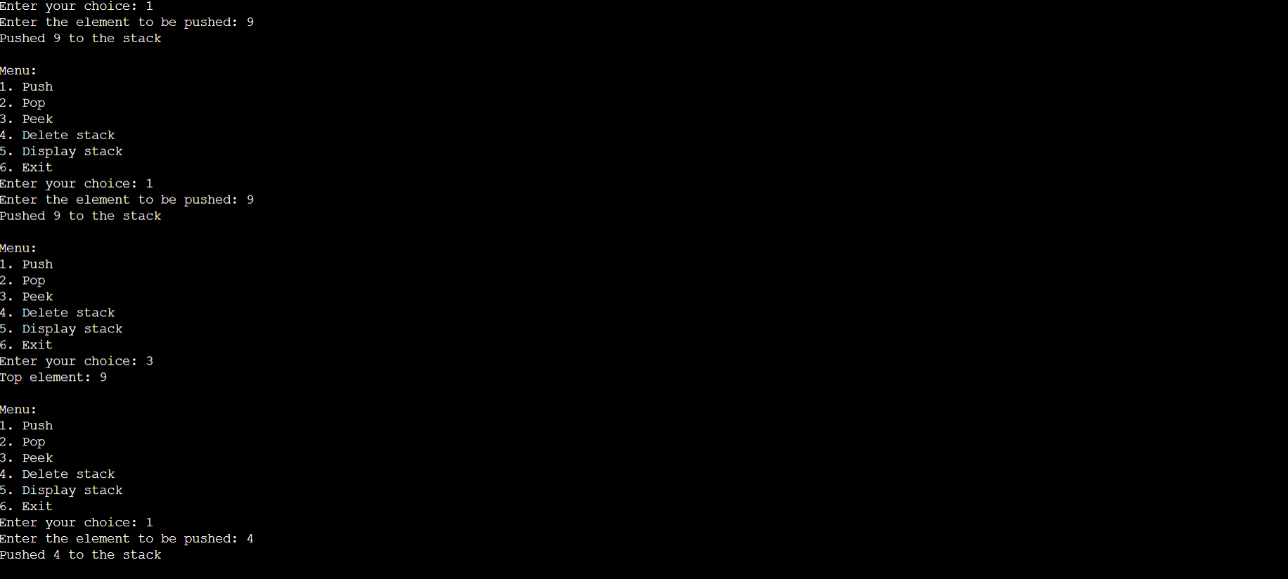
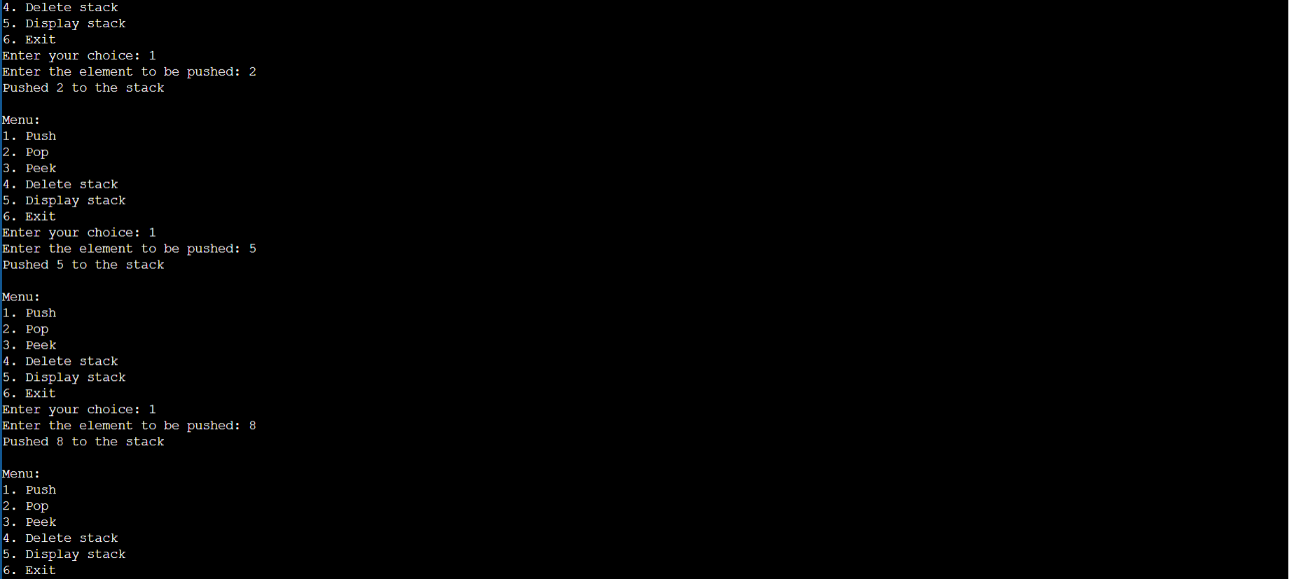
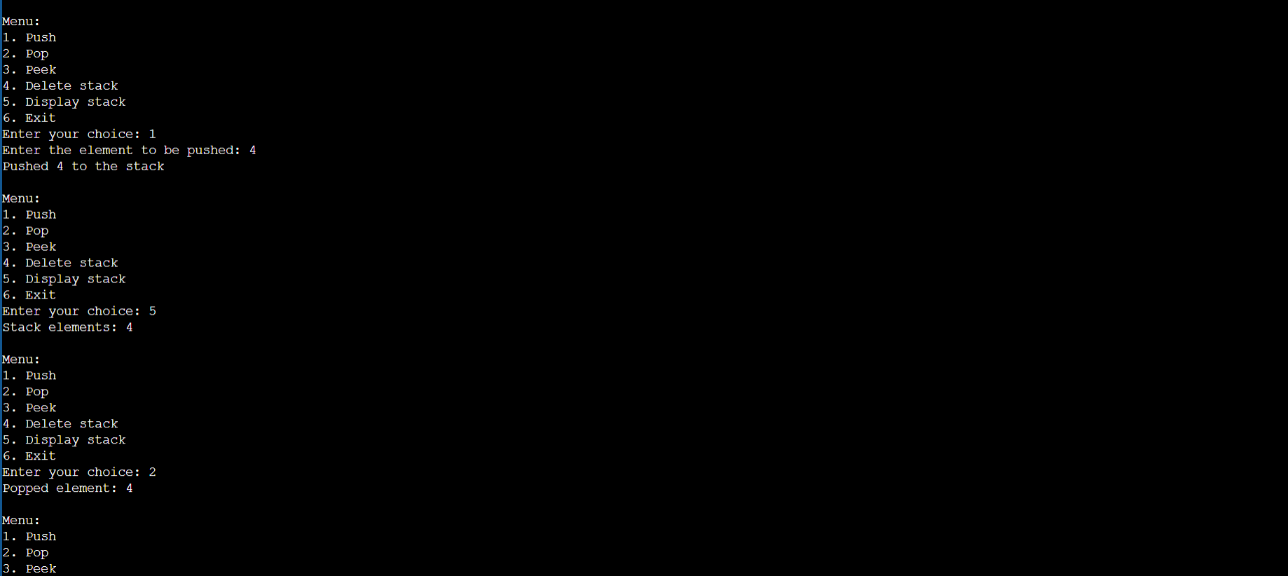
}

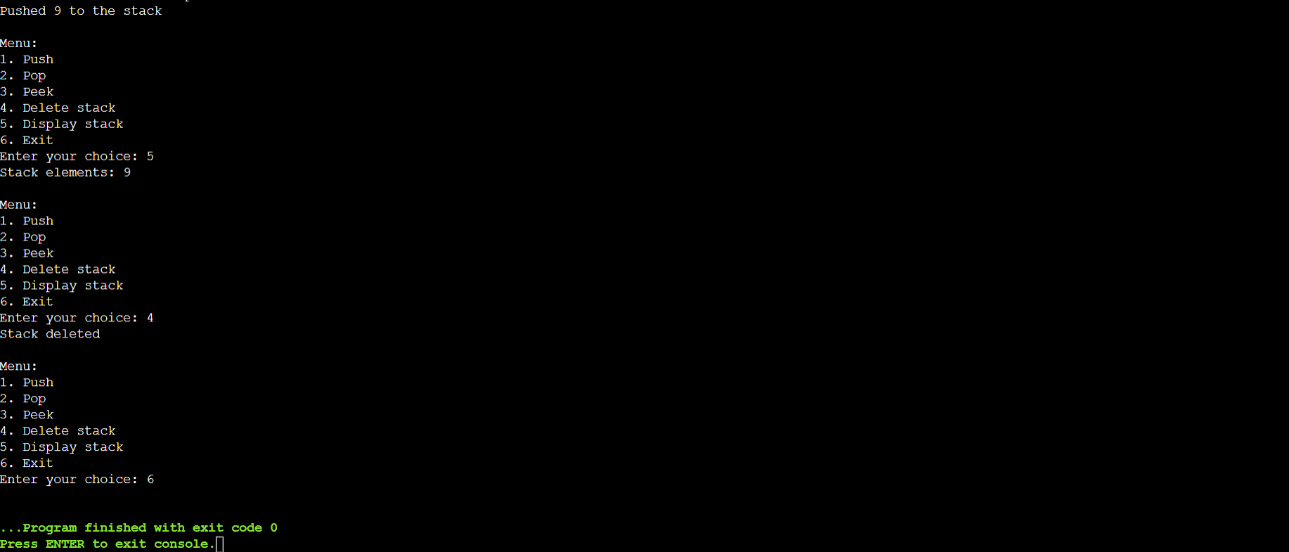
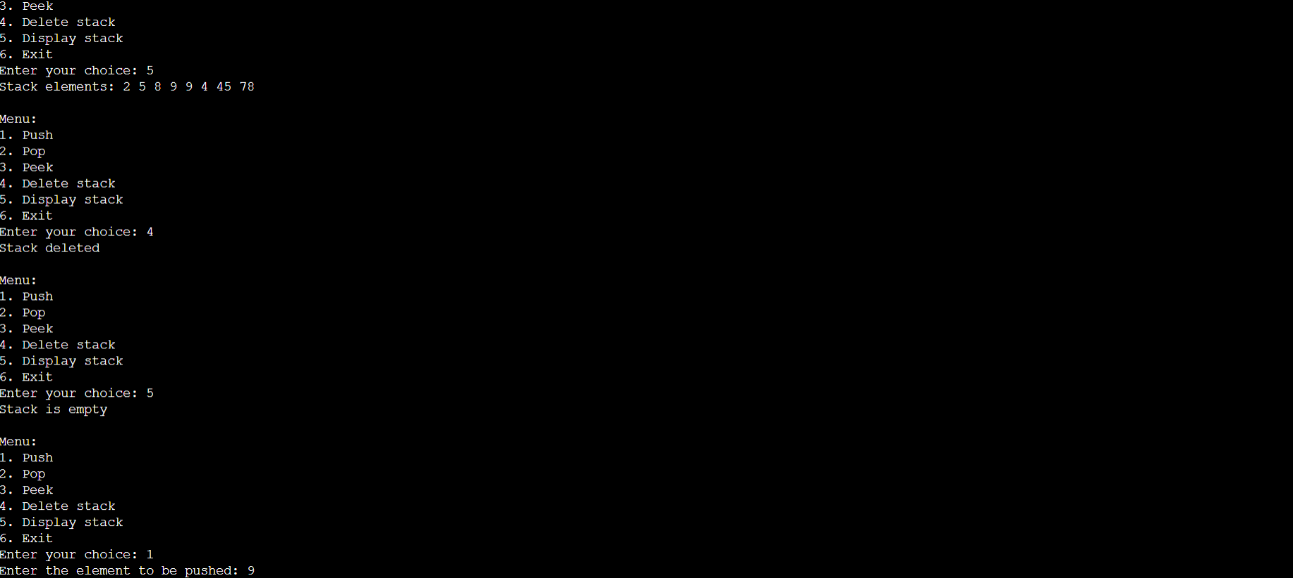
}

return 0;

}

**Output Screenshots:**

****

****

**PostLab Questions:**

1. **List 5 Applications of Stack Data Structures.**

**Ans:**

1. Function Calls-

The state of the program is placed into the Stack when a function is invoked. The preceding function's execution is continued after the process returns by popping the state off the Stack.

2. Parenthesis checking-

To determine if brackets are balanced or not, a stack data structure is utilized. An opening parenthesis is popped off the Stack as a closing parenthesis is added onto it. The brackets are balanced if the Stack is empty at the conclusion of the expression.

3. Undo/Redo Operations-

Many apps' undo-redo functionality employs stacks to remember the prior operations. A new action is added to the Stack each time it is completed. The top member of the Stack is popped to undo the action, and the original procedure is then carried out.

4. Web browser history-

Stacks are used by web browsers to record the websites you visit. When you click the back button, the previous URL is removed from the Stack and is added to the Stack each time you visit a new page.

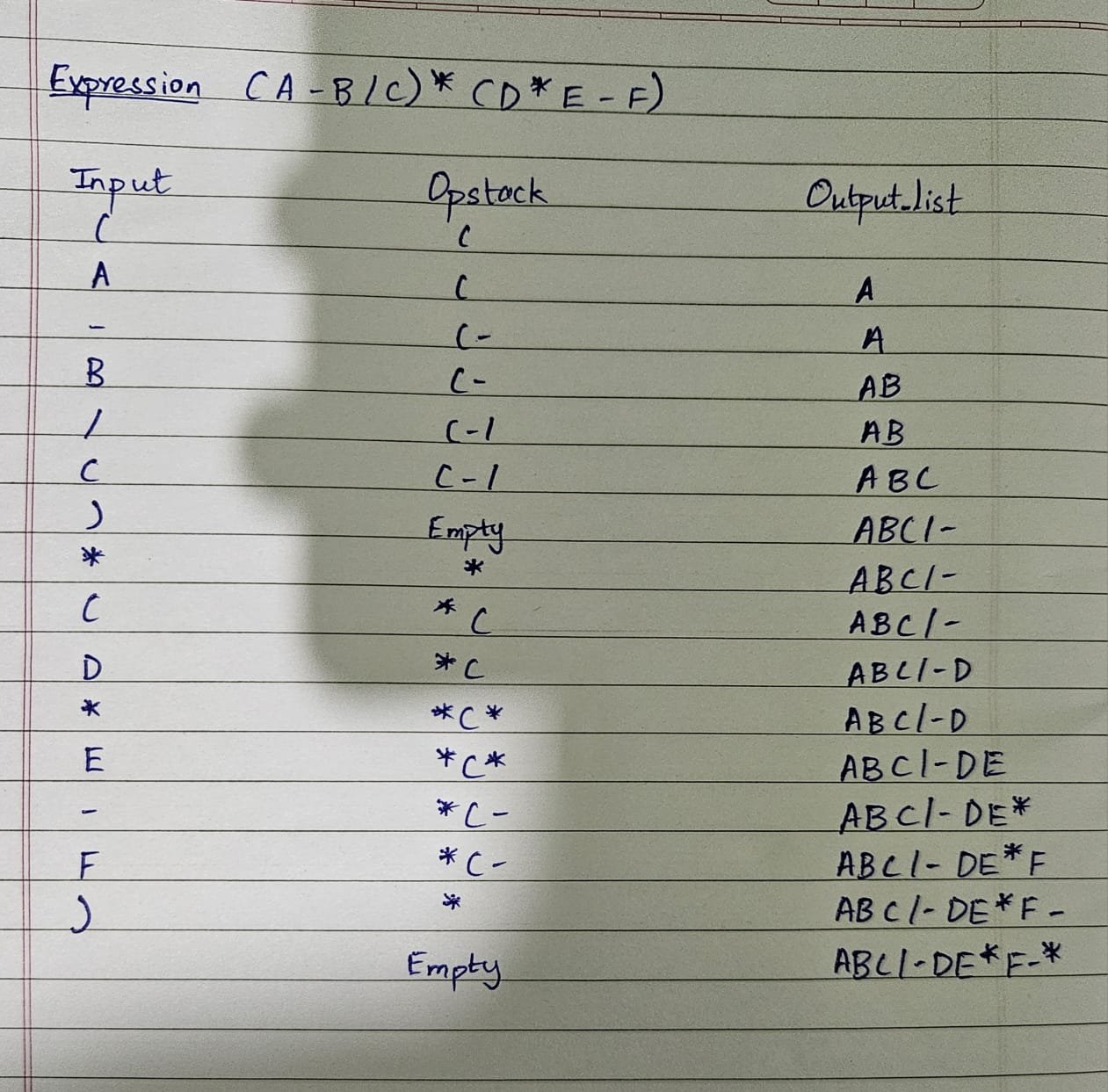
5. Reverse the Data-

We must reorganize the data so that the first and final items are switched; the second and second-last elements are exchanged, and so on for all subsequent elements if we want to reverse a particular collection of data.

1. **Convert the given Infix Expression into Postfix Expression using Stack:**

**(A-B/C)\*(D\*E-F)**

**Ans:**



1. **Explain How stack can be used in both Nested Function calls and Recursion using suitable examples for each. Further Define Activation Records used for Function Calling.**

**Ans. Not done in class**

**Conclusion:-**

In this experiment, we implemented stack adt using array. Also we learnt about the various uses of stack in day to day programming. We understood the working behind the creation of stack, insertion, deletion and displaying the element in the stack. Also we understood how to check if the stack is full or empty using the topelement index