| **Part A**  **Name:- Aryan Srivastava**  **Roll No:- A073**  **Subject:- Data Structures and Algorithms**  **Program: B Tech/MBA Tech CE 2nd Year** |
| --- |
| **Aim:**  To study and implement concept of Stack data structure |
| **Prerequisite:** C++ Programming |
| **Outcome:** To implement stack and its operations. |
| * **Theory:** Stack is a linear data structure which stores the elements in an ordered manner. * The elements in a stack are added and removed only from one end which is called top. * The policy is LIFO, the element that was inserted last is the first one to be taken out.   Operations-   1. Push(element)- inserts an element at top of the stack 2. Pop() – removes the topmost element 3. Peek()- returns the topmost element without removing 4. isEmpty() – checks if stack is empty 5. isFull()- checks if stack is full   Push operation   * Step 1: IF TOP == MAX-1   + PRINT OVERFLOW   + Goto step 4 * [END OF IF] * Step 2: SET TOP = TOP + 1 * Step 3: SET STACK[TOP] = VALUE * Step 4: END   Pop operation   * Step 1: IF TOP = -1   + PRINT UNDERFLOW   + Go to step 4 * [END OF IF] * Step 2: SET VAL = STACK[TOP] * Step 3: SET TOP = TOP - 1 * Step 4: END   Peek operation   * Step 1: IF TOP = -1   + PRINT UNDERFLOW   + Go to step 4 * [END OF IF] * Step 2: SET VAL = STACK[TOP] * Step 3: SET TOP = TOP - 1 * Step 4: END |
| **Procedure:**   1. Open CodeBlock editor or visual studio editor and write the code in C++. 2. Complile and run the code |
| **Instructions:**   1. Copy code & paste in code section and output of Part B. |
|  |

| **Part B** |
| --- |
| **Code:**  **import java.util.\*;**  **class stacks {**  **int[] stackArray;**  **int top;**  **int capacity;**  **stacks(int capacity) {**  **this.capacity = capacity;**  **this.stackArray = new int[capacity];**  **this.top = -1;**  **}**  **void push(int element) {**  **if (top == capacity - 1) {**  **System.out.println("Stack is full. Cannot push element.");**  **} else {**  **stackArray[++top] = element;**  **System.out.println("Pushed: " + element);**  **}**  **}**  **int pop() {**  **if (top == -1) {**  **System.out.println("Stack is empty. Cannot pop element.");**  **return -1;**  **} else {**  **return stackArray[top--];**  **}**  **}**  **int top() {**  **if (top == -1) {**  **System.out.println("Stack is empty. No top element.");**  **return -1;**  **} else {**  **return stackArray[top];**  **}**  **}**  **public static void main(String[] args) {**  **stacks stack = new stacks(5);**    **stack.push(10);**  **stack.push(20);**  **stack.push(30);**  **stack.push(40);**  **stack.push(50);**  **stack.push(60);**  **System.out.println("Top element: " + stack.top());**  **System.out.println("Popped element: " + stack.pop());**  **System.out.println("Popped element: " + stack.pop());**  **System.out.println("Popped element: " + stack.pop());**  **System.out.println((stack.top()==-1)? "stack is empty": ("Top element is: "+stack.top()));**  **System.out.println("Popped element: " + stack.pop());**  **System.out.println("Popped element: " + stack.pop());**  **System.out.println((stack.top()==-1)? "stack is empty": ("Top element is: "+stack.top()));**  **}**  **}** |
| **Output:**  **Pushed: 10**  **Pushed: 20**  **Pushed: 30**  **Pushed: 40**  **Pushed: 50**  **Stack is full. Cannot push element.**  **Top element: 50**  **Popped element: 50**  **Popped element: 40**  **Popped element: 30**  **Top element is: 20**  **Popped element: 20**  **Popped element: 10**  **Stack is empty. No top element.**  **stack is empty** |
| **Observation & Learning:**  Write your Observations & Learning after performing task |
| **Answer Following Question** |
| **Conclusion:**  We successfully implemented Java program on arrays and its operations. |