PRACTICAL – 4

PROGRAM -1

AIM- WRITE A C program PUSH Operations

CODE-

|  |
| --- |
| #include <stdio.h>  #define MAX 10 /\* Defining the maximum size of the stack \*/  int stack[MAX], top = -1;  // Function to add an element to the stack  void push(int value) {  if (top == MAX - 1) {  printf("Stack Overflow\n");  } else {  top = top + 1;  stack[top] = value;  }  }  // Function to display the stack  void display() {  int i;  if (top == -1) {  printf("Stack is empty\n");  } else {  printf("Stack elements are:\n");  for (i = top; i >= 0; i--) {  printf("%d\n", stack[i]);  }  }  }  int main() {  printf("\nHARSH D\n");  int value, choice;  do {  printf("\nChoose an option:\n");  printf("1. Push element\n");  printf("2. Display stack\n");  printf("3. Exit\n");  scanf("%d", &choice);  switch (choice) {  case 1:  printf("Enter element to push onto the stack: ");  scanf("%d", &value);  push(value);  break;  case 2:  display();  break;  case 3:  printf("Exiting...\n");  break;  default:  printf("Invalid choice. Please try again.\n");  break; }  } while (choice != 3);  return 0;  } |

OUTPUT-

|  |
| --- |
|  |

PROGRAM -2

AIM- WRITE A C program POP Operations

CODE-

|  |
| --- |
| // Stack opeartions pop  #include <stdio.h>  #include <stdlib.h> // For exit function  #define MAX\_SIZE 10  int stack[MAX\_SIZE];  int top = -1; // Initialize top index to -1 (empty stack)  // Function to check if the stack is empty  int isEmpty() {  return top == -1;  }  // Function to check if the stack is full  int isFull() {  return top == MAX\_SIZE - 1;  }  // Function to push an element onto the stack  void push(int value) {  if (isFull()) {  printf("Stack Overflow: Cannot push element\n");  exit(EXIT\_FAILURE); // Terminate program on stack overflow  }  stack[++top] = value; // Increment top and add value to stack  printf("Pushed %d onto the stack\n", value);  }  // Function to pop an element from the stack  int pop() {  if (isEmpty()) {  printf("Stack Underflow: Stack is empty\n");  exit(EXIT\_FAILURE); // Terminate program on stack underflow  }  int value = stack[top--]; // Get value from top and decrement top  printf("Popped %d from the stack\n", value);  return value;  }  // Function to display the stack elements  void display() {  if (isEmpty()) {  printf("Stack is empty\n");  return;  }  printf("Stack elements are:\n");  for (int i = top; i >= 0; i--) {  printf("%d\n", stack[i]);  }  }  int main() {  printf("\n HARSH D \n");  // Push some elements onto the stack  push(10);  push(20);  push(30);  display();  // Pop an element from the stack  int poppedValue = pop();  printf("Popped value: %d\n", poppedValue);  display();  return 0;} |

OUTPUT-

|  |
| --- |
|  |

PROGRAM -3

AIM- WRITE A C program IsEmpty Operations

CODE-

|  |
| --- |
| // Stack opeartions IsEmpty  #include <stdio.h>  #define MAXSIZE 10 // Define the maximum size of the stack  int stack[MAXSIZE];  int top = -1;  // Function to check if the stack is empty  int IsEmpty() {  if(top == -1)  return 1; // The stack is empty  else  return 0; // The stack is not empty  }  // Function to push an element to the stack  void push(int data) {  if(top == MAXSIZE - 1) {  printf("Stack Overflow\n");  } else {  stack[++top] = data;  }  }  // Function to pop an element from the stack  int pop() {  if(IsEmpty()) {  printf("Stack Underflow\n");  return -1;  } else {  return stack[top--];  }  }  int main()  {  printf("\n HARSH D \n");  // Push elements to the stack  push(10);  push(20);  push(30);  // Check if the stack is empty  if(IsEmpty()) {  printf("The stack is empty.\n");  } else {  printf("The stack is not empty.\n");  }  // Pop all elements to empty the stack  pop();  pop();  pop();  // Check again if the stack is empty  if(IsEmpty()) {  printf("The stack is now empty.\n");  } else {  printf("The stack is still not empty.\n");  }  return 0;  } |

OUTPUT:-

|  |
| --- |
|  |

PROGRAM - 4

AIM- WRITE A C program IsFull Operations

CODE-

|  |
| --- |
| // Stack opeartions IsFull  #include <stdio.h>  #define MAXSIZE 10 // Define the maximum size of the stack  int stack[MAXSIZE];  int top = -1;  // Function to check if the stack is full  int IsFull() {  if(top == MAXSIZE - 1)  return 1; // The stack is full  else  return 0; // The stack is not full  }  // Function to push an element to the stack  void push(int data) {  if(IsFull()) {  printf("Stack Overflow\n");  } else {  stack[++top] = data;  }  }  // Function to display the stack  void display() {  if(top == -1) {  printf("Stack is empty\n");  } else {  printf("Stack elements are:\n");  for(int i = top; i >= 0; i--)  printf("%d\n", stack[i]);  }  }  int main()  {  printf("\n HARSH D \n")  // Push elements to the stack until it is full  for(int i = 0; i < MAXSIZE; i++) {  push(i);  }  // Check if the stack is full  if(IsFull()) {  printf("The stack is full.\n");  } else {  printf("The stack is not full.\n");  }  // Display the stack  display();  return 0;  } |

OUTPUT-

|  |
| --- |
|  |

PROGRAM - 5

AIM- WRITE A C program TOP Operations

CODE-

|  |
| --- |
| // Stack opeartions TOP  #include <stdio.h>  #define MAXSIZE 10 // Define the maximum size of the stack  int stack[MAXSIZE];  int top = -1;  // Function to check if the stack is empty  int isEmpty() {  return top == -1;  }  // Function to return the top element of the stack  int topOperation() {  if (!isEmpty()) {  return stack[top];  } else {  printf("Stack is empty\n");  return -1; // Return -1 if the stack is empty  }  }  // Function to push an element to the stack  void push(int data) {  if (top == MAXSIZE - 1) {  printf("Stack Overflow\n");  } else {  stack[++top] = data;  }  }  // Function to display the current elements of the stack  void display()  {  int i;  if (isEmpty()) {  printf("Stack is empty\n");  } else {  printf("Stack elements are:\n");  for ( i = top; i >= 0; i--) {  printf("%d\n", stack[i]);  }  }  }  int main()  {  printf("\n HARSH D \n");  // Push elements to the stack  push(10);  push(20);  push(30);  // Perform the TOP operation  int topElement = topOperation();  if (topElement != -1) {  printf("Top element is: %d\n", topElement);  }  // Display the current elements of the stack  display();  return 0;  } |

OUTPUT-

|  |
| --- |
|  |