Rajalakshmi Engineering College

Name: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 15

Section 1 : MCQ

1. Consider the linked list implementation of a stack.

Which of the following nodes is considered as Top of the stack?

Answer

Last node

Status : Wrong ***Marks : 0/1***

2. What is the value of the postfix expression 6 3 2 4 + - *?

Answer

-18

Status : Correct ***Marks : 1/1***

3. A user performs the following operations on stack of size 5 then which of the following is correct statement for Stack?

push(1);
pop();
push(2);
push(3);
pop();
push(2);
pop();
pop();
push(4);
pop();
pop();
push(5);

Answer

Underflow Occurs

Status : Correct ***Marks : 1/1***

4. What is the primary advantage of using an array-based stack with a fixed size?

Answer

Efficient memory usage

Status : Correct ***Marks : 1/1***

5. Pushing an element into the stack already has five elements. The stack size is 5, then the stack becomes

Answer

Overflow

Status : Correct ***Marks : 1/1***

6. The result after evaluating the postfix expression 10 5 + 60 6 / * 8 - is

Answer

142

Status : Correct ***Marks : 1/1***

7. Which of the following operations allows you to examine the top element of a stack without removing it?

Answer

Peek

Status : Correct ***Marks : 1/1***

8. What is the advantage of using a linked list over an array for implementing a stack?

Answer

Linked lists can dynamically resize

Status : Correct ***Marks : 1/1***

9. When you push an element onto a linked list-based stack, where does the new element get added?

Answer

At the end of the list

Status : Wrong ***Marks : 0/1***

10. Which of the following Applications may use a Stack?

Answer

All of the mentioned options

Status : Correct ***Marks : 1/1***

11. What will be the output of the following code?

```
#include <stdio.h>
#define MAX_SIZE 5
int stack[MAX_SIZE];
int top = -1;
int isEmpty() {
   return (top == -1);
}
int isFull() {
   return (top == MAX_SIZE - 1);
}
void push(int item) {
   if (isFull())
      printf("Stack Overflow\n");
   else
      stack[++top] = item;
}
int main() {
   printf("%d\n", isEmpty());
   push(10);
   push(20);
   push(30);
   printf("%d\n", isFull());
   return 0;
}
```

Answer

10

Status : Correct ***Marks : 1/1***

12. Consider a linked list implementation of stack data structure with three operations:

push(value): Pushes an element value onto the stack.pop(): Pops the top element from the stack.top(): Returns the item stored at the top of the stack.

Given the following sequence of operations:

push(10);pop();push(5);top();

What will be the result of the stack after performing these operations?

Answer

The top element in the stack is 5

Status : Correct ***Marks : 1/1***

13. The user performs the following operations on the stack of size 5 then at the end of the last operation, the total number of elements present in the stack is

```
push(1);
pop();
push(2);
push(3);
pop();
push(4);
pop();
pop();
push(5);
```

Answer

1

Status : Correct ***Marks : 1/1***

14. Here is an Infix Expression: 4+3*(6*3-12). Convert the expression from Infix to Postfix notation. The maximum number of symbols that will appear on the stack AT ONE TIME during the conversion of this expression?

Answer

3

Status : Wrong ***Marks : 0/1***

15. In an array-based stack, which of the following operations can result

in a Stack underflow?

Answer

Popping an element from an empty stack

Status : Correct ***Marks : 1/1***

16. In the linked list implementation of the stack, which of the following operations removes an element from the top?

Answer

Pop

Status : Correct ***Marks : 1/1***

17. What will be the output of the following code?

```
#include <stdio.h>
#define MAX_SIZE 5
void push(int* stack, int* top, int item) {
    if (*top == MAX_SIZE - 1) {
        printf("Stack Overflow\n");
        return;
    }
    stack[++(*top)] = item;
}
int pop(int* stack, int* top) {
    if (*top == -1) {
        printf("Stack Underflow\n");
        return -1;
    }
    return stack[(*top)--];
}

int main() {
    int stack[MAX_SIZE];
    int top = -1;
    push(stack, &top, 10);
```

```
    push(stack, &top, 20);
    push(stack, &top, 30);
    printf("%d\n", pop(stack, &top));
    printf("%d\n", pop(stack, &top));
    printf("%d\n", pop(stack, &top));
    printf("%d\n", pop(stack, &top));
    return 0;
}
```

Answer

302010Stack Underflow

Status : Wrong ***Marks : 0/1***

18. What will be the output of the following code?

```
#include <stdio.h>
#define MAX_SIZE 5
int stack[MAX_SIZE];
int top = -1;
void display() {
    if (top == -1) {
        printf("Stack is empty\n");
    } else {
        printf("Stack elements: ");
        for (int i = top; i >= 0; i--) {
            printf("%d ", stack[i]);
        }
        printf("\n");
    }
}
void push(int value) {
    if (top == MAX_SIZE - 1) {
        printf("Stack Overflow\n");
    } else {
        stack[++top] = value;
    }
}
int main() {
```

```
    display();
    push(10);
    push(20);
    push(30);
    display();
    push(40);
    push(50);
    push(60);
    display();
    return 0;
}
```

Answer

Stack is emptyStack elements: 30 20 10Stack OverflowStack elements: 50 40 30 20 10

Status : Correct ***Marks : 1/1***

19. Elements are Added on _____ of the Stack.

Answer

Top

Status : Correct ***Marks : 1/1***

20. In a stack data structure, what is the fundamental rule that is followed for performing operations?

Answer

First In First Out

Status : Wrong ***Marks : 0/1***

Rajalakshmi Engineering College

Name: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

In a coding competition, you are assigned a task to create a program that simulates a stack using a linked list.

The program should feature a menu-driven interface for pushing an integer to stack, popping, and displaying stack elements, with robust error handling for stack underflow situations. This challenge tests your data structure skills.

Input Format

The input consists of integers corresponding to the operation that needs to be performed:

Choice 1: Push the integer value onto the stack. If the choice is 1, the following input is a space-separated integer, representing the element to be pushed onto

the stack.

Choice 2: Pop the integer from the stack.

Choice 3: Display the elements in the stack.

Choice 4: Exit the program.

Output Format

The output displays messages according to the choice and the status of the stack:

If the choice is 1, push the given integer to the stack and display the following: "Pushed element: " followed by the value pushed.

If the choice is 2, pop the integer from the stack and display the following: "Popped element: " followed by the value popped.

If the choice is 2, and if the stack is empty without any elements, print "Stack is empty. Cannot pop."

If the choice is 3, print the elements in the stack: "Stack elements (top to bottom): " followed by the space-separated values.

If the choice is 3, and there are no elements in the stack, print "Stack is empty".

If the choice is 4, exit the program and display the following: "Exiting program".

If any other choice is entered, print "Invalid choice".

Refer to the sample input and output for the exact format.

Sample Test Case

Input: 1 3
1 4
3
2
3
4
Output: Pushed element: 3
Pushed element: 4
Stack elements (top to bottom): 4 3
Popped element: 4
Stack elements (top to bottom): 3
Exiting program

Answer

-

Status : Skipped ***Marks : 0/10***

Rajalakshmi Engineering College

Name: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Sanjeev is in charge of managing a library's book storage, and he wants to create a program that simplifies this task. His goal is to implement a program that simulates a stack using an array.

Help him in writing a program that provides the following functionality:

Add Book ID to the Stack (Push): You can add a book ID to the top of the book stack. Remove Book ID from the Stack (Pop): You can remove the top book ID from the stack and display its details. If the stack is empty, you cannot remove any more book IDs.Display Books ID in the Stack (Display): You can view the books ID currently on the stack.Exit the Library: You can choose to exit the program.

Input Format

The input consists of integers corresponding to the operation that needs to be performed:

Choice 1: Push the book onto the stack. If the choice is 1, the following input is a space-separated integer, representing the ID of the book to be pushed onto the stack.

Choice 2: Pop the book ID from the stack.

Choice 3: Display the book ID in the stack.

Choice 4: Exit the program.

Output Format

The output displays messages according to the choice and the status of the stack:

1. If the choice is 1, push the given book ID to the stack and display the corresponding message.
2. If the choice is 2, pop the book ID from the stack and display the corresponding message.
3. If the choice is 2, and if the stack is empty without any book ID, print "Stack Underflow"
4. If the choice is 3, print the book IDs in the stack.
5. If the choice is 3, and there are book IDs in the stack, print "Stack is empty"
6. If the choice is 4, exit the program and display the corresponding message.
7. If any other choice is entered, print "Invalid choice"

Refer to the sample output for the exact text and format.

Sample Test Case

Input: 1 19
1 28
2
3
2
4
Output: Book ID 19 is pushed onto the stack
Book ID 28 is pushed onto the stack

Book ID 28 is popped from the stack
Book ID in the stack: 19
Book ID 19 is popped from the stack
Exiting the program

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

#define MAX_SIZE 100

struct Stack {
    int data[MAX_SIZE];
    int top;
};

void initializeStack(struct Stack *stack) {
    stack->top = -1;
}

int isEmpty(struct Stack *stack) {
    return stack->top == -1;
}

int isFull(struct Stack *stack) {
    return stack->top == MAX_SIZE - 1;
}

void push(struct Stack *stack, int bookID) {
    if (isFull(stack)) {
        printf("Stack Overflow\n");
        return;
    }

    stack->data[++stack->top] = bookID;
    printf("Book ID %d is pushed onto the stack\n", bookID);
}

void pop(struct Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack Underflow\n");
```

```
        return;
    }

    int bookID = stack->data[stack->top--];
    printf("Book ID %d is popped from the stack\n", bookID);
}

void display(struct Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty\n");
        return;
    }

    printf("Book ID in the stack: ");
    for (int i = stack->top; i >= 0; i--) {
        printf("%d ", stack->data[i]);
    }
    printf("\n");
}

int main() {
    struct Stack stack;
    initializeStack(&stack);

    int choice, bookID;

    while (1) {
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                scanf("%d", &bookID);
                push(&stack, bookID);
                break;

            case 2:
                pop(&stack);
                break;

            case 3:
                display(&stack);
                break;
```

```
            case 4:
                printf("Exiting the program\n");
                return 0;

            default:
                printf("Invalid choice\n");
        }
    }

    return 0;
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

Sharon is developing a programming challenge for a coding competition. The challenge revolves around implementing a character-based stack data structure using an array.

Sharon's project involves a stack that can perform the following operations:

Push a Character: Users can push a character onto the stack.Pop a Character: Users can pop a character from the stack, removing and displaying the top character.Display Stack: Users can view the current elements in the stack.Exit: Users can exit the stack operations application.

Write a program to help Sharon to implement a program that performs the given operations.

Input Format

The input consists of integers corresponding to the operation that needs to be performed:

Choice 1: Push the character onto the stack. If the choice is 1, the following input is a space-separated character, representing the character to be pushed onto the stack.

Choice 2: Pop the character from the stack.

Choice 3: Display the characters in the stack.

Choice 4: Exit the program.

Output Format

The output displays messages according to the choice and the status of the stack:

1. If the choice is 1, push the given character to the stack and display the pushed character having the prefix "Pushed: ".
2. If the choice is 2, undo the character from the stack and display the character that is popped having the prefix "Popped: ".
3. If the choice is 2, and if the stack is empty without any characters, print "Stack is empty. Nothing to pop."
4. If the choice is 3, print the elements in the stack having the prefix "Stack elements: ".
5. If the choice is 3, and there are no characters in the stack, print "Stack is empty."
6. If the choice is 4, exit the program.
7. If any other choice is entered, print "Invalid choice"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2
4

Output: Stack is empty. Nothing to pop.

Answer

-

Status : Skipped *Marks : 0/10*

Rajalakshmi Engineering College

Name: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

You are a software developer tasked with building a module for a scientific calculator application. The primary function of this module is to convert infix mathematical expressions, which are easier for users to read and write, into postfix notation (also known as Reverse Polish Notation). Postfix notation is more straightforward for the application to evaluate because it removes the need for parentheses and operator precedence rules.

The scientific calculator needs to handle various mathematical expressions with different operators and ensure the conversion is correct. Your task is to implement this infix-to-postfix conversion algorithm using a stack-based approach.

Example

Input:

a+b

Output:

ab+

Explanation:

The postfix representation of (a+b) is ab+.

Input Format

The input is a string, representing the infix expression.

Output Format

The output displays the postfix representation of the given infix expression.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: a+(b*e)
Output: abe*+

Answer

-

Status : Skipped ***Marks : 0/10***

Rajalakshmi Engineering College

Name: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

Milton is a diligent clerk at a school who has been assigned the task of managing class schedules. The school has various sections, and Milton needs to keep track of the class schedules for each section using a stack-based system.

He uses a program that allows him to push, pop, and display class schedules for each section. Milton's program uses a stack data structure, and each class schedule is represented as a character. Help him write a program using a linked list.

Input Format

The input consists of integers corresponding to the operation that needs to be performed:

Choice 1: Push the character onto the stack. If the choice is 1, the following input is a space-separated character, representing the class schedule to be pushed onto the stack.

Choice 2: Pop class schedule from the stack

Choice 3: Display the class schedules in the stack.

Choice 4: Exit the program.

Output Format

The output displays messages according to the choice and the status of the stack:

- If the choice is 1, push the given class schedule to the stack and display the following: "Adding Section: [class schedule]"
- If the choice is 2, pop the class schedule from the stack and display the following: "Removing Section: [class schedule]"
- If the choice is 2, and if the stack is empty without any class schedules, print "Stack is empty. Cannot pop."
- If the choice is 3, print the class schedules in the stack in the following: "Enrolled Sections: " followed by the class schedules separated by space.
- If the choice is 3, and there are no class schedules in the stack, print "Stack is empty"
- If the choice is 4, exit the program and display the following: "Exiting the program"
- If any other choice is entered, print "Invalid choice"

Refer to the sample output for the exact format.

Sample Test Case

Input: 1 d
1 h
3
2

3
4
Output: Adding Section: d
Adding Section: h
Enrolled Sections: h d
Removing Section: h
Enrolled Sections: d
Exiting program

Answer

-

Status : Skipped ***Marks : 0/10***

Rajalakshmi Engineering College

Name: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Suppose you are building a calculator application that allows users to enter mathematical expressions in infix notation. One of the key features of your calculator is the ability to convert the entered expression to postfix notation using a Stack data structure.

Write a function to convert infix notation to postfix notation using a Stack.

Input Format

The input consists of a string, an infix expression that includes only digits(0-9), and operators(+, -, *, /).

Output Format

The output displays the equivalent postfix expression of the given infix expression.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1+2*3/4-5

Output: 123*4/+5-

Answer

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define MAX_SIZE 100

typedef struct {
    int top;
    char items[MAX_SIZE];
} Stack;

void initStack(Stack *s) {
    s->top = -1;
}

int isEmpty(Stack *s) {
    return s->top == -1;
}

void push(Stack *s, char item) {
    if (s->top == MAX_SIZE - 1) {
        printf("Stack Overflow\n");
        return;
    }
    s->items[++(s->top)] = item;
}

char pop(Stack *s) {
    if (isEmpty(s)) {
        printf("Stack Underflow\n");
```

```
        return -1;
    }
    return s->items[(s->top)--];
}

char peek(Stack *s) {
    if (isEmpty(s)) {
        return -1;
    }
    return s->items[s->top];
}

int isOperator(char c) {
    return (c == '+' || c == '-' || c == '*' || c == '/');
}

int precedence(char c) {
    if (c == '*' || c == '/')
        return 2;
    else if (c == '+' || c == '-')
        return 1;
    else
        return 0;
}

void infixToPostfix(char *infix, char *postfix) {
    Stack s;
    initStack(&s);

    int i, j = 0;
    char c;

    for (i = 0; infix[i] != '\0'; i++) {
        c = infix[i];

        if (isdigit(c)) {
            postfix[j++] = c;
        }
        else if (isOperator(c)) {
            while (!isEmpty(&s) && precedence(peek(&s)) >= precedence(c)) {
                postfix[j++] = pop(&s);
            }
```

```
            push(&s, c);
        }
        else if (c == '(') {
            push(&s, c);
        }
        else if (c == ')') {
            while (!isEmpty(&s) && peek(&s) != '(') {
                postfix[j++] = pop(&s);
            }

            if (!isEmpty(&s) && peek(&s) == '(') {
                pop(&s);
            }
        }
    }

    while (!isEmpty(&s)) {
        postfix[j++] = pop(&s);
    }

    postfix[j] = '\0';
}

int main() {
    char infix[MAX_SIZE];
    char postfix[MAX_SIZE];

    fgets(infix, MAX_SIZE, stdin);

    size_t len = strlen(infix);
    if (len > 0 && infix[len-1] == '\n') {
        infix[len-1] = '\0';
    }

    infixToPostfix(infix, postfix);

    printf("%s\n", postfix);

    return 0;
}
```

Status : Correct **Marks : 10/10**

2. Problem Statement

You are required to implement a stack data structure using a singly linked list that follows the Last In, First Out (LIFO) principle.

The stack should support the following operations: push, pop, display, and peek.

Input Format

The input consists of four space-separated integers N, representing the elements to be pushed onto the stack.

Output Format

The first line of output displays all four elements in a single line separated by a space.

The second line of output is left blank to indicate the pop operation without displaying anything.

The third line of output displays the space separated stack elements in the same line after the pop operation.

The fourth line of output displays the top element of the stack using the peek operation.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 11 22 33 44
Output: 44 33 22 11

33 22 11
33

Answer

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

```
struct Node {
    int data;
    struct Node* next;
};

struct Stack {
    struct Node* top;
};

struct Stack* createStack() {
    struct Stack* stack = (struct Stack*)malloc(sizeof(struct Stack));
    stack->top = NULL;
    return stack;
}

int isEmpty(struct Stack* stack) {
    return (stack->top == NULL);
}

void push(struct Stack* stack, int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Stack Overflow\n");
        return;
    }

    newNode->data = value;
    newNode->next = stack->top;
    stack->top = newNode;
}

int pop(struct Stack* stack) {
    if (isEmpty(stack)) {
        printf("Stack Underflow\n");
        return -1;
    }

    struct Node* temp = stack->top;
    int popped = temp->data;
    stack->top = stack->top->next;
    free(temp);
```

```
    return popped;
}

int peek(struct Stack* stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty\n");
        return -1;
    }
    return stack->top->data;
}

void display(struct Stack* stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty\n");
        return;
    }

    struct Node* temp = stack->top;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    struct Stack* stack = createStack();
    int num1, num2, num3, num4;

    scanf("%d %d %d %d", &num1, &num2, &num3, &num4);

    push(stack, num1);
    push(stack, num2);
    push(stack, num3);
    push(stack, num4);

    display(stack);

    printf("\n");

    pop(stack);
```

```
    display(stack);

    printf("%d\n", peek(stack));

    return 0;
}
```

Status : Correct **Marks : 10/10**

3. Problem Statement

Buvi is working on a project that requires implementing an array-stack data structure with an additional feature to find the minimum element.

Buvi needs to implement a program that simulates a stack with the following functionalities:

Push: Adds an element onto the stack.Pop: Removes the top element from the stack.Find Minimum: Finds the minimum element in the stack.

Buvi's implementation should efficiently handle these operations with a maximum stack size of 20.

Input Format

The first line of input consists of an integer N, representing the number of elements to push onto the stack.

The second line consists of N space-separated integer values, representing the elements to be pushed onto the stack.

Output Format

The first line of output displays "Minimum element in the stack: " followed by the minimum element in the stack after pushing all elements.

The second line displays "Popped element: " followed by the popped element.

The third line displays "Minimum element in the stack after popping: " followed by the minimum element in the stack after popping one element.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
5 2 8 1

Output: Minimum element in the stack: 1
Popped element: 1
Minimum element in the stack after popping: 2

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

#define MAX_SIZE 20

struct MinStack {
    int stack[MAX_SIZE];
    int minStack[MAX_SIZE];
    int top;
    int minTop;
};

void initialize(struct MinStack *ms) {
    ms->top = -1;
    ms->minTop = -1;
}

int isEmpty(struct MinStack *ms) {
    return (ms->top == -1);
}

int isFull(struct MinStack *ms) {
    return (ms->top == MAX_SIZE - 1);
}

void push(struct MinStack *ms, int value) {
    if (isFull(ms)) {
        printf("Stack Overflow\n");
```

```
        return;
    }

    ms->top++;
    ms->stack[ms->top] = value;

    if (ms->minTop == -1 || value <= ms->minStack[ms->minTop]) {
        ms->minTop++;
        ms->minStack[ms->minTop] = value;
    }
}

int pop(struct MinStack *ms) {
    if (isEmpty(ms)) {
        printf("Stack Underflow\n");
        return -1;
    }

    int popped = ms->stack[ms->top];
    ms->top--;

    if (popped == ms->minStack[ms->minTop]) {
        ms->minTop--;
    }

    return popped;
}

int findMin(struct MinStack *ms) {
    if (isEmpty(ms)) {
        printf("Stack is empty\n");
        return -1;
    }

    return ms->minStack[ms->minTop];
}

int main() {
    struct MinStack ms;
    initialize(&ms);

    int n, value;
```

```
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        scanf("%d", &value);
        push(&ms, value);
    }

    printf("Minimum element in the stack: %d\n", findMin(&ms));

    int popped = pop(&ms);
    printf("Popped element: %d\n", popped);

    printf("Minimum element in the stack after popping: %d\n", findMin(&ms));

    return 0;
}
```

Status : Correct **Marks : 10/10**