Rajalakshmi Engineering College

Name: Manisha S
Email: 240801191@rajalakshmi.edu.in
Roll no: 240801191
Phone: 8667375390
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

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

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

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

// Define the structure of a stack node
typedef struct Node {
    int data;
    struct Node* next;
} Node;

// Function to push an element onto the stack
void push(Node** top, int value) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) {
        return;
    }
    newNode->data = value;
    newNode->next = *top;
    *top = newNode;
}
```

```
// Function to pop an element from the stack
void pop(Node** top) {
    if (*top == NULL) {
        return;
    }
    Node* temp = *top;
    *top = (*top)->next;
    free(temp);
}

// Function to display the stack
void display(Node* top) {
    while (top != NULL) {
        printf("%d ", top->data);
        top = top->next;
    }
    printf("\n");
}

// Function to peek (view the top element)
int peek(Node* top) {
    if (top != NULL) {
        return top->data;
    }
    return -1;  // Return -1 if stack is empty
}

int main() {
    Node* top = NULL;
    int N[4];

    // Read four space-separated integers
    for (int i = 0; i < 4; i++) {
        scanf("%d", &N[i]);
        push(&top, N[i]);  // Push elements onto the stack
    }

    // Display the stack
    display(top);

    // Pop the top element
```

```
    pop(&top);

    // Display the stack after pop operation
    printf("\n");  // Blank line for pop operation
    display(top);

    // Peek and display the top element
    printf("%d\n", peek(top));

    return 0;
}
```

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

2. 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>

#define MAX_SIZE 20

// Function to find the minimum element in the stack
int findMin(int stack[], int size) {
    int min = stack[0];
    for (int i = 1; i < size; i++) {
        if (stack[i] < min) {
            min = stack[i];
        }
    }
    return min;
}

int main() {
    int N;
    scanf("%d", &N);

    // Validate input constraints
    if (N < 2 || N > MAX_SIZE) {
        return 0;
    }
```

```
int stack[MAX_SIZE];
int top = -1;

// Push elements onto the stack
for (int i = 0; i < N; i++) {
    int value;
    scanf("%d", &value);
    stack[++top] = value;
}

// Find and display the minimum element in the stack
printf("Minimum element in the stack: %d\n", findMin(stack, N));

    // Pop the top element and display it
    int poppedElement = stack[top--];
    printf("Popped element: %d\n", poppedElement);

    // Find and display the minimum element after popping
    printf("Minimum element in the stack after popping: %d\n", findMin(stack, top
+ 1));

    return 0;
}
```

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

3. Problem Statement

Rithi is building a simple text editor that allows users to type characters, undo their typing, and view the current text. She has implemented this text editor using an array-based stack data structure.

She has to develop a basic text editor with the following features:

Type a Character (Push): Users can type a character and add it to the text editor.Undo Typing (Pop): Users can undo their typing by removing the last character they entered from the editor.View Current Text (Display): Users can view the current text in the editor, which is the sequence of characters in the buffer.Exit: Users can exit the text editor application.

Write a program that simulates this text editor's undo feature using a

character stack and implements the push, pop and display operations accordingly.

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, print: "Typed character: <character>" where <character> is the character that was pushed to the stack.
2. If the choice is 2, print: "Undo: Removed character <character>" where <character> is the character that was removed from the stack.
3. If the choice is 2, and if the stack is empty without any characters, print "Text editor buffer is empty. Nothing to undo."
4. If the choice is 3, print: "Current text: <character1> <character2> ... <characterN>" where <character1>, <character2>, ... are the characters in the stack, starting from the last pushed character.
5. If the choice is 3, and there are no characters in the stack, print "Text editor buffer 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: 1 H
1 A
3
4

Output: Typed character: H
Typed character: A
Current text: A H

Answer

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

#define MAX_SIZE 100

char stack[MAX_SIZE];
int top = -1;

void push(char ch) {
    if (top < MAX_SIZE - 1) {
        stack[++top] = ch;
        printf("Typed character: %c\n", ch);
    } else {
        printf("Stack is full, cannot type more characters.\n");
    }
}

void pop() {
    if (top >= 0) {
        printf("Undo: Removed character %c\n", stack[top--]);
    } else {
        printf("Text editor buffer is empty. Nothing to undo.\n");
    }
}


void display() {
    if (top >= 0) {
        printf("Current text:");
        for (int i = top; i >= 0; i--) {
            printf(" %c", stack[i]);
        }
```

```
        printf("\n");
    } else {
        printf("Text editor buffer is empty.\n");
    }
}

int main() {
    int choice;
    char ch;

    while (1) {
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                scanf(" %c", &ch);
                push(ch);
                break;
            case 2:
                pop();
                break;
            case 3:
                display();
                break;
            case 4:
                return 0;
            default:
                printf("Invalid choice\n");
        }
    }
    return 0;
}
```

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