Rajalakshmi Engineering College

Name: Phaveen S
Email: 240701383@rajalakshmi.edu.in
Roll no: 240701383
Phone: null
Branch: REC
Department: I CSE FD
Batch: 2028
Degree: B.E - CSE

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

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>
#include <string.h>

#define MAX_SIZE 100

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

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

void pop() {
    if (top == -1) {
        printf("Text editor buffer is empty. Nothing to undo.\n");
    } else {
        char removed = stack[top--];
        printf("Undo: Removed character %c\n", removed);
    }
}

void display() {
    if (top == -1) {
        printf("Text editor buffer is empty.\n");
    } else {
        printf("Current text: ");
        for (int i = top; i >= 0; i--) {
            printf("%c ", stack[i]);
        }
```

```
        printf("\n");
    }
}

int main() {
    int choice;
    char ch;

    while (1) {
        if (scanf("%d", &choice) != 1) {
            while (getchar() != '\n');
            printf("Invalid choice\n");
            continue;
        }

        switch (choice) {
            case 1:
                if (scanf(" %c", &ch) == 1) {
                    push(ch);
                }
                break;
            case 2:
                pop();
                break;
            case 3:
                display();
                break;
            case 4:
                // Exit
                exit(0);
                break;
            default:
                printf("Invalid choice\n");
        }
    }

    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

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

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

// Global top pointer
struct Node* top = NULL;

void push(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = top;
    top = newNode;
}

void pop() {
    if (top != NULL) {
        struct Node* temp = top;
        top = top->next;
        free(temp);
    }
}

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

int peek() {
    if (top != NULL)
        return top->data;
    return -1;
}

int main() {
    int a, b, c, d;
```

```
    scanf("%d %d %d %d", &a, &b, &c, &d);

    push(a);
    push(b);
    push(c);
    push(d);

    display();
    printf("\n");

    pop();

    display();
    printf("\n");

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

    return 0;
}
```

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

3. Problem Statement

In an educational setting, Professor Smith tasks Computer Science students with designing an algorithm to evaluate postfix expressions efficiently, fostering problem-solving skills and understanding of stack-based computations.

The program prompts users to input a postfix expression, evaluates it, and displays the result, aiding students in honing their coding abilities.

Input Format

The input consists of the postfix mathematical expression.

The expression will contain real numbers and mathematical operators (+, -, *, /), without any space.

Output Format

The output prints the result of evaluating the given postfix expression.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 82/

Output: 4

Answer

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

#define MAX 100

float stack[MAX];
int top = -1;

void push(float value) {
    if (top >= MAX - 1) {
        printf("Stack overflow\n");
        return;
    }
    stack[++top] = value;
}

float pop() {
    if (top == -1) {
        printf("Stack underflow\n");
        exit(1);
    }
    return stack[top--];
}

float evaluatePostfix(char* expr) {
    int i = 0;
    char ch;

    while ((ch = expr[i++]) != '\0') {
```

```
        if (isdigit(ch)) {
            push((float)(ch - '0'));
        } else {
            float val2 = pop();
            float val1 = pop();
            switch (ch) {
                case '+': push(val1 + val2); break;
                case '-': push(val1 - val2); break;
                case '*': push(val1 * val2); break;
                case '/': push(val1 / val2); break;
                default:
                    printf("Invalid operator: %c\n", ch);
                    exit(1);
            }
        }
    }

    return pop();
}

int main() {
    char expr[100];
    scanf("%s", expr);

    float result = evaluatePostfix(expr);

    if (result == (int)result)
        printf("%d\n", (int)result);
    else
        printf("%.2f\n", result);

    return 0;
}
```

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