Rajalakshmi Engineering College

Name: Pooja S
Email: 240701386@rajalakshmi.edu.in
Roll no: 240701386
Phone: 8838480229
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 : 2
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>

#define MAX 100

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

// Push operation
void push(char ch) {
    if (top < MAX - 1) {
        stack[++top] = ch;
        printf("Typed character: %c\n", ch);
    } else {
        printf("Text editor buffer is full.\n");
    }
}

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

// Display operation
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");
    }
}

// Main function
int main() {
    int choice;
    char ch;

    while (1) {
        if (scanf("%d", &choice) != 1) break;

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

    return 0;
}
```

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

2. Problem Statement

Siri is a computer science student who loves solving mathematical problems. She recently learned about infix and postfix expressions and was fascinated by how they can be used to evaluate mathematical expressions.

She decided to write a program to convert an infix expression with operators to its postfix form. Help Siri in writing the program.

Input Format

The input consists of a single line containing an infix expression.

Output Format

The output prints a single line containing the postfix expression equivalent to the given infix expression.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: (2 + 3) * 4

Output: 23+4*

Answer

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

#define MAX 100

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

// Push to stack
void push(char ch) {
    if (top < MAX - 1)
        stack[++top] = ch;
}

// Pop from stack
char pop() {
    if (top >= 0)
```

```
        return stack[top--];
    return '\0';
}

// Peek top of stack
char peek() {
    if (top >= 0)
        return stack[top];
    return '\0';
}

// Check if character is operator
int isOperator(char ch) {
    return ch == '+' || ch == '-' || ch == '*' || ch == '/';
}

// Return precedence of operators
int precedence(char op) {
    if (op == '+' || op == '-') return 1;
    if (op == '*' || op == '/') return 2;
    return 0;
}

int main() {
    char infix[100], postfix[100];
    int i = 0, k = 0;

    // Read input line
    fgets(infix, sizeof(infix), stdin);

    while (infix[i] != '\0' && infix[i] != '\n') {
        char ch = infix[i];

        if (isdigit(ch)) {
            postfix[k++] = ch;
        }
        else if (ch == ' ') {
            // Skip spaces
        }
        else if (ch == '(') {
            push(ch);
        }
        else if (ch == ')') {
            while (top != -1 && peek() != '(') {
                postfix[k++] = pop();
            }
            pop(); // Remove '('
        }
        else if (isOperator(ch)) {
            while (top != -1 && precedence(peek()) >= precedence(ch)) {
                postfix[k++] = pop();
            }
            push(ch);
        }

        i++;
    }

    // Pop remaining operators
    while (top != -1) {
        postfix[k++] = pop();
    }

    postfix[k] = '\0';
    printf("%s\n", postfix);

    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>

#define MAX 20

int stack[MAX], minStack[MAX];
int top = -1, minTop = -1;

// Push function with min tracking
void push(int x) {
    if (top < MAX - 1) {
        stack[++top] = x;
        if (minTop == -1 || x <= minStack[minTop]) {
            minStack[++minTop] = x;
        }
    }
}

// Pop function with min tracking
int pop() {
    if (top == -1) return -1;

    int popped = stack[top--];
    if (popped == minStack[minTop]) {
        minTop--;
    }
    return popped;
}

// Get current minimum
int getMin() {
    if (minTop == -1) return -1;
    return minStack[minTop];
}

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

    for (i = 0; i < N; i++) {
        scanf("%d", &val);
        push(val);
    }

    printf("Minimum element in the stack: %d\n", getMin());
    int popped = pop();
    printf("Popped element: %d\n", popped);
    printf("Minimum element in the stack after popping: %d\n", getMin());

    return 0;
}
```

Status : Correct *Marks : 10/10*