Rajalakshmi Engineering College

Name: bharath kumar
Email: 241801032@rajalakshmi.edu.in
Roll no: 2116241801032
Phone: 7305320010
Branch: REC
Department: I AI & DS FA
Batch: 2028
Degree: B.E - AI & DS

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

Latha is taking a computer science course and has recently learned about infix and postfix expressions. She is fascinated by the idea of converting infix expressions into postfix notation. To practice this concept, she wants to implement a program that can perform the conversion for her.

Help Latha by designing a program that takes an infix expression as input and outputs its equivalent postfix notation.

Example

Input:

(3+4)5

Output:

34+5

Input Format

The input consists of a string, the infix expression to be converted to postfix notation.

Output Format

The output displays a string, the postfix expression equivalent of the input infix expression.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: A+B*C-D/E

Output: ABC*+DE/-

Answer

```
// You are using GCC
#include<stdio.h>
#include<string.h>
#include<ctype.h>
#define s 100
char box[s];
int t=-1;
void push(char c){
   box[++t]=c;
}
char pop(){
   return box[t--];
}
int precedence(char op){
   switch(op){
      case '+':
      case '-':return 1;
      case '*':
      case '/':return 2;
      default:return 0;
   }
```

```
}
void intopost(char *in){
    char post[s];
    int i,k=0;
    char c;
    for(i=0;in[i];i++){
        c= in[i];
        if(isalnum(c)){
            post[k++]=c;
        }
        else if(c=='('){
            push(c);
        }
        else if(c==')'){
            while(t!=-1 && box[t]!='('){
                post[k++]=pop();
            }pop();
        }
        else{
            while(t!=-1 && precedence(box[t])>=precedence(c)){
                post[k++]=pop();
            }
            push(c);
        }
    }
    while(t!=-1){
        post[k++]=pop();
    }
    post[k]='\0';
    printf("%s",post);
}
int main(){
    char in[s];
    scanf("%s",in);
    intopost(in);
    return 0;
}
```

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

2. 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

struct Stack {
   double items[MAX];
   int top;
};

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

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

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

void push(struct Stack* s, double value) {
    if (!isFull(s)) {
        s->items[++(s->top)] = value;
    }
}

double pop(struct Stack* s) {
    if (!isEmpty(s)) {
        return s->items[(s->top)--];
    }
    return 0;
}

double evaluatePostfix(const char* expression) {
    struct Stack s;
    initStack(&s);
    for (int i = 0; expression[i] != '\0'; i++) {
        if (isdigit(expression[i])) {
            push(&s, expression[i] - '0');
        } else {
            double operand2 = pop(&s);
            double operand1 = pop(&s);
            switch (expression[i]) {
                case '+':
                    push(&s, operand1 + operand2);
                    break;
                case '-':
                    push(&s, operand1 - operand2);
                    break;
                case '*':
                    push(&s, operand1 * operand2);
                    break;
                case '/':
                    push(&s, operand1 / operand2);
                    break;
            }
        }
    }
    return pop(&s);
}

int main() {
    char expression[MAX];
    scanf("%s", expression);
    double result = evaluatePostfix(expression);
    printf("%.0f\n", result);
    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>
#include <string.h>

#define MAX_SIZE 100

typedef struct {
   char stack[MAX_SIZE];
   int top;
} TextEditor;

void init(TextEditor *editor) {
   editor->top = -1;
}

int isFull(TextEditor *editor) {
   return editor->top == MAX_SIZE - 1;
}

int isEmpty(TextEditor *editor) {
   return editor->top == -1;
}

void push(TextEditor *editor, char character) {
   if (!isFull(editor)) {
      editor->stack[++editor->top] = character;
      printf("Typed character: %c\n", character);
   }
}

char pop(TextEditor *editor) {
   if (!isEmpty(editor)) {
      return editor->stack[editor->top--];
   }
   return '\0';
}

void display(TextEditor *editor) {
   if (!isEmpty(editor)) {
      printf("Current text: ");
```

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

int main() {
    TextEditor editor;
    init(&editor);
    int choice;
    char character;

    while (1) {
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                scanf(" %c", &character);
                push(&editor, character);
                break;
            case 2:
                if (!isEmpty(&editor)) {
                    character = pop(&editor);
                    printf("Undo: Removed character %c\n", character);
                } else {
                    printf("Text editor buffer is empty. Nothing to undo.\n");
                }
                break;
            case 3:
                display(&editor);
                break;
            case 4:
                exit(0);
            default:
                printf("Invalid choice\n");
        }
    }
    return 0;
}
```

Status : Correct *Marks : 10/10*