Rajalakshmi Engineering College

Name: Subashri .S
Email: 240801336@rajalakshmi.edu.in
Roll no: 240801336
Phone: 9080419390
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

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

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

#define MAX 100

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

void initStack(Stack *s) {
    s->top = -1;
}
void push(Stack *s, char c) {
    if (s->top == MAX - 1) {
        printf("Stack overflow\n");
        return;
    }
    s->items[++(s->top)] = c;
```

```
}

char pop(Stack *s) {
    if (s->top == -1) {
        return '\0';
    }
    return s->items[(s->top)--];
}

char peek(Stack *s) {
    if (s->top == -1) {
        return '\0';
    }
    return s->items[s->top];
}

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

int isOperator(char c) {
    return (c == '+' || c == '-' || c == '*' || c == '/');
}
void infixToPostfix(char *infix, char *postfix) {
    Stack s;
    initStack(&s);
    int i = 0, j = 0;

    while (infix[i] != '\0') {
        if (isalnum(infix[i])) {
            postfix[j++] = infix[i];
        } else if (infix[i] == '(') {
            push(&s, infix[i]);
        } else if (infix[i] == ')') {
            while (peek(&s) != '(') {
                postfix[j++] = pop(&s);
            }
            pop(&s);
        } else if (isOperator(infix[i])) {
            while (s.top != -1 && precedence(peek(&s)) >= precedence(infix[i])) {
```

```
            postfix[j++] = pop(&s);
        }
        push(&s, infix[i]);
    }
    i++;
  }

  while (s.top != -1) {
    postfix[j++] = pop(&s);
  }

  postfix[j] = '\0';
}

int main() {
  char infix[MAX], postfix[MAX];
  scanf("%s", infix);

  infixToPostfix(infix, postfix);

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

  return 0;
}
```

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

2. 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 <ctype.h>
#include <string.h>
#define MAX 30
typedef struct {
    char items[MAX];
    int top;
} 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 - 1) {
        printf("Stack overflow\n");
        return;
    }
    s->items[++s->top] = item;
}

char pop(Stack *s) {
    if (isEmpty(s)) {
        printf("Stack underflow\n");
        return '\0';
```

```
    }
    return s->items[s->top--];
}
int precedence(char op) {
    switch (op) {
        case '+':
        case '-': return 1;
        case '*':
        case '/': return 2;
        default: return 0;
    }
}
void infixToPostfix(char *infix, char *postfix) {
    Stack s;
    initStack(&s);
    int i, j = 0;
    char ch;

    for (i = 0; infix[i] != '\0'; i++) {
        ch = infix[i];
        if (isdigit(ch)) {
            postfix[j++] = ch;
        }
        else if (ch == '(') {
            push(&s, ch);
        }
        else if (ch == ')') {
            while (!isEmpty(&s) && s.items[s.top] != '(') {
                postfix[j++] = pop(&s);
            }
            pop(&s);
        }
        else {
            while (!isEmpty(&s) && precedence(s.items[s.top]) >= precedence(ch)) {
                postfix[j++] = pop(&s);
            }
            push(&s, ch);
        }
    }

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

```
    }
    postfix[j] = '\0';
}

int main() {
    char infix[MAX], postfix[MAX];

    scanf("%s", infix);

    infixToPostfix(infix, postfix);

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

    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 100
typedef struct {
    char items[MAX];
    int top;
}Stack;
void initStack(Stack *s) {
    s->top = -1;
}
int isEmpty(Stack *s) {
    return s->top == -1;
}

void push(Stack *s, char ch) {
    if (s->top == MAX - 1) {
        printf("Stack overflow\n");
        return;
    }
    s->items[++s->top] = ch;
    printf("Typed character: %c\n", ch);
}
void pop(Stack *s) {
    if (isEmpty(s)) {
        printf("Text editor buffer is empty. Nothing to undo.\n");
        return;
    }
    printf("Undo: Removed character %c\n", s->items[s->top--]);
}

void display(Stack *s) {
    if (isEmpty(s)) {
        printf("Text editor buffer is empty.\n");
        return;
    }
    printf("Current text: ");
    for (int i = s->top; i >= 0; i--) {
        printf("%c ", s->items[i]);
```

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

int main() {
    Stack s;
    initStack(&s);
    int choice;
    char ch;

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

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

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