

# Rajalakshmi Engineering College

Name: Nivedhitha K  
Email: 240701371@rajalakshmi.edu.in  
Roll no: 240701371  
Phone: 9790413580  
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

Raj is a software developer, and his team is building an application that processes user inputs in the form of strings containing brackets. One of the essential features of the application is to validate whether the input string meets specific criteria.

During testing, Raj inputs the string "([()]){}". The application correctly returns "Valid string" because the input satisfies the criteria: every opening bracket (, [, and { has a corresponding closing bracket ), ], and }, arranged in the correct order.

Next, Raj tests the application with the string "(])". This time, the application correctly returns "Invalid string" because the opening bracket [ is incorrectly closed by the bracket ), which violates the validation rules.

Finally, Raj enters the string "{[()]}" . The application correctly identifies it as a "Valid string" since all opening brackets are matched with the corresponding closing brackets in the correct order.

As a software developer, Raj's responsibility is to ensure that the application works reliably and produces accurate results for all input strings, following the validation rules. He accomplishes this by using a method for solving such problems.

### ***Input Format***

The input comprises a string representing a sequence of brackets that need to be validated.

### ***Output Format***

The output prints "Valid string" if the string is valid. Otherwise, it prints "Invalid string".

Refer to the sample output for formatting specifications.

### ***Sample Test Case***

Input: ({[()]})

Output: Valid string

### ***Answer***

```
// You are using GCC
#include <stdio.h>
#include <string.h>
#define MAX 100
char stack[MAX];
int top=-1;
void push(char ch)
{
    if (top<MAX-1)
    {
        stack[++top]=ch;
    }
}
```

```

char pop()
{
    if (top>=0)
    {
        return stack[top--];
    }
    return '\0';
}

int isMatchingPair(char open, char close)
{
    return (open=='(' && close==')') || (open=='{' && close=='}') || (open=='[' &&
close==']');
}

int isValidString(char *str)
{
    for (int i=0; str[i]!='\0'; i++)
    {
        char ch=str[i];
        if (ch=='(' || ch=='{' || ch=='[')
        {
            push(ch);
        }
        else if (ch==')' || ch=='}' || ch==']')
        {
            char open=pop();
            if (!isMatchingPair(open, ch))
            {
                return 0;
            }
        }
    }
    return top==-1;
}

int main()
{
    char str[MAX];
    scanf("%s", str);
    if (isValidString(str))
    {
        printf("Valid string\n");
    }
    else

```

```
{  
    printf("Invalid string\n");  
}  
return 0;  
}
```

**Status :** Correct

**Marks :** 10/10

## 2. 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 <stdlib.h>
#include <string.h>
#include <ctype.h>
#define MAX 100
typedef struct {
    int top;
    char items[MAX];
} Stack;
void initStack(Stack* stack)
{
    stack->top=-1;
}
int isEmpty(Stack* stack)
{
    return stack->top== -1;
}
void push(Stack* stack, char item)
{
    if (stack->top<MAX-1)
    {
        stack->items[++stack->top]=item;
    }
}
char pop(Stack* stack)
{
    if (!isEmpty(stack))
    {
        return stack->items[stack->top--];
    }
    return '\0';
}
char peek(Stack* stack)
{
    if (!isEmpty(stack))
```

```

    {
        return stack->items[stack->top];
    }
    return '\0';
}
int precedence(char operators)
{
    switch (operators)
    {
        case '+':
        case '-':
            return 1;
        case '*':
        case '/':
            return 2;
        case '^':
            return 3;
        default:
            return 0;
    }
}
void infixToPostfix(char* infix, char* postfix)
{
    Stack stack;
    initStack(&stack);
    int j=0;
    for (int i=0; infix[i]; i++)
    {
        char c=infix[i];
        if (isalnum(c))
        {
            postfix[j++]=c;
        }
        else if (c=='(')
        {
            push(&stack, c);
        }
        else if (c==')')
        {
            while (!isEmpty(&stack) && peek(&stack)!='(')
            {
                postfix[j++]=pop(&stack);
            }
        }
    }
    postfix[j]='\0';
}

```

```

    }
    pop(&stack);
}
else
{
    while (!isEmpty(&stack) && precedence(peek(&stack))>=precedence(c))
    {
        postfix[j++]=pop(&stack);
    }
    push(&stack, c);
}
}
while (!isEmpty(&stack))
{
    postfix[j++]=pop(&stack);
}
postfix[j]='\0';
}
int main()
{
    char infix[MAX];
    char postfix[MAX];
    printf("");
    fgets(infix, sizeof(infix), stdin);
    infix[strcspn(infix, "\n")]=0;
    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>
```

```
#define MAX 1000
```

```
char stack[MAX];
```

```
int top=-1;
```

```
void push(char ch)
```

```
{
```

```
    if (top==MAX-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 ch=stack[top--];
```

```
        printf("Undo: Removed character %c\n", ch);
    }
}
```

```
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)
    {
        scanf("%d", &choice);
        switch(choice)
        {
            case 1:
                scanf(" %c", &ch);
                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