Rajalakshmi Engineering College

Name: pooja D
Email: 240701385@rajalakshmi.edu.in
Roll no: 240701385
Phone: 9677250159
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

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

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

int precedence(char op) {
    switch (op) {
        case '+':
        case '-': return 1;
        case '*':
        case '/': return 2;
        default: return 0;
    }
}

void infixToPostfix(char *exp) {
    char stack[30], postfix[60];
    int top = -1, i = 0, k = 0;
    char ch;

    while ((ch = exp[i++]) != '\0') {
        if (isdigit(ch)) {
            postfix[k++] = ch;
        } else if (ch == '(') {
            stack[++top] = ch;
        } else if (ch == ')') {
            while (top != -1 && stack[top] != '(') {
                postfix[k++] = stack[top--];
            }
            if (top != -1) top--;
        } else {
            while (top != -1 && precedence(ch) <= precedence(stack[top]) &&
```

240701385

```
stack[top] != '(') {
            postfix[k++] = stack[top--];
         }
         stack[++top] = ch;
      }
   }

   while (top != -1) {
      postfix[k++] = stack[top--];
   }

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

int main() {
   char exp[31];
   if (fgets(exp, sizeof(exp), stdin)) {
      infixToPostfix(exp);
   }
   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

240701385

given infix expression.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: (2 + 3) * 4
Output: 23+4*

Answer

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

int precedence(char op) {
   switch (op) {
      case '+':
      case '-': return 1;
      case '*':
      case '/': return 2;
      default: return 0;
   }
}

void infixToPostfix(char *exp) {
   char stack[50], postfix[100];
   int top = -1, i = 0, k = 0;
   char ch;

   while ((ch = exp[i++]) != '\0') {
      if (ch == ' ') continue;
      if (isdigit(ch)) {
         postfix[k++] = ch;
      } else if (ch == '(') {
         stack[++top] = ch;
      } else if (ch == ')') {
         while (top != -1 && stack[top] != '(') {
            postfix[k++] = stack[top--];
         }
         if (top != -1) top--;
```

240701385

```
        } else {
            while (top != -1 && precedence(ch) <= precedence(stack[top]) &&
stack[top] != '(') {
                postfix[k++] = stack[top--];
            }
            stack[++top] = ch;
        }
    }

    while (top != -1) {
        postfix[k++] = stack[top--];
    }

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

int main() {
    char exp[51];
    if (fgets(exp, sizeof(exp), stdin)) {
        infixToPostfix(exp);
    }
    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.

240701385

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;

void push(int val) {
    if (top < MAX - 1) {
        stack[++top] = val;
        if (minTop == -1 || val <= minStack[minTop]) {
            minStack[++minTop] = val;
        }
    }
}

int pop() {
    if (top == -1) return -1;
    int popped = stack[top--];
    if (popped == minStack[minTop]) {
        minTop--;
    }
    return popped;
}

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

    return 0;
}
```

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

240701385 240701385 240701385 240701385