**Topic – Evaluation of a Postfix Expression**

* **Problem Statement**

**Write a C program to evaluate a postfix expression. Note that each operand is 1 digit length.**

**Input example:**

Enter the postfix expression

93/5\*653-\*+862-/-

**Output example:**

The result of this expression is 25.0000

* **Proposed C Code**

**/\* ---------- postfix.c--------------- \*/**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<string.h>**

**#define SIZE 100**

**/\* Stack size is taken as 100 \*/**

**static int top=-1;/\* global top variable which is used for push and pop \*/**

**void stackempty()**

**{**

**printf("Stack is empty\n");**

**exit(0);**

**}**

**void stackfull()**

**{**

**printf("Stack is full\n");**

**exit(0);**

**}**

**/\* function to push item \*/**

**void push(float\* stack,float item)**

**{**

**if(top>=SIZE-1)**

**stackfull();**

**else**

**stack[++top] = item;**

**}**

**/\* function to pop \*/**

**float pop(float \*stack)**

**{**

**if(top==-1)**

**stackempty();**

**else**

**return stack[top--];**

**}**

**int main()**

**{**

**float operation = 0,operand1 = 0,operand2 = 0;/\* variables needed for calculation \*/**

**int val = 0;/\* variable to scan digit from input string \*/**

**char \*str = (char\*)malloc(100\*sizeof(char));/\* input string of postfix expression \*/**

**printf("Enter the postfix expression\n");**

**scanf("%s",str);**

**float\* stack = (float\*)malloc(SIZE\*sizeof(float));/\* stack formation \*/**

**for ( int i = 0 ; i < strlen(str) ; i++ )**

**{**

**/\* pushing the digits when operand is found \*/**

**if ((str[i]>='0')&&(str[i]<='9'))**

**{**

**val = str[i]-48;**

**push(stack,val);**

**}**

**/\* if an operator is found required number of operands are poped \*/**

**/\* After performing operation between them,the result is pushed back \*/**

**else if( str[i]=='+')**

**{**

**operand2 = (float)pop(stack);**

**operand1 = (float)pop(stack);**

**operation = operand1 + operand2;**

**push(stack,operation);**

**}**

**else if( str[i]=='\*')**

**{**

**operand2 = (float)pop(stack);**

**operand1 = (float)pop(stack);**

**operation = operand1 \* operand2;**

**push(stack,operation);**

**}**

**else if( str[i]=='-')**

**{**

**operand2 = (float)pop(stack);**

**operand1 = (float)pop(stack);**

**operation = operand1 - operand2;**

**push(stack,operation);**

**}**

**else if( str[i]=='/')**

**{**

**operand2 = (float)pop(stack);**

**operand1 = (float)pop(stack);**

**operation = operand1 / operand2;**

**push(stack,operation);**

**}**

**}**

**/\* After all the operations stack[top] contains the final value \*/**

**printf("The result of this expression is %0.4f\n",stack[top]);**

**free(str); /\* freeing the memory after output is generated \*/**

**free(stack);**

**return 0;**

**}**

**/\*------------------------------------------------------------------------------------------------------------------------- \*/**

* **Conclusion**

**The proposed algorithm has overall runtime of O(n) where n is size of the string.**

* **Limitations : This algorithm will work for the expression for which each operand is of single digit .**
* **Assumptions: Here we are considering the postfix expression is valid and the total operands are (total operators + 1) .**