**Infix to Postfix**

Problem Statement:

Given a valid infix expression ( each operand is one digit), write a C program to convert the infix expression into a postfix expression.

• Input example :

1. 6/2 -3+4\*2
2. ((6/2) -3)+4\*2
3. a+b\*(c^d-e)^(f+g\*h)-i

• Output example :

1. 62/3-42\*+
2. 62/3-42\*+
3. abcd^e-fgh\*+^\*+i-

Proposed C Code:

/\* ------- main.c ------- \*/

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

char \*stack;

int top=-1;

int size=100000;

void init(){

stack=(char\*)calloc(size,sizeof(char));

}

void push(char item){

stack[++top]=item;

}

char pop(){

return stack[top--];

}

char peek(){

return stack[top];

}

int isEmpty(){

if(top==-1){

return 1;

}else{

return 0;

}

}

int prec(char ch)

{

switch (ch) {

case '+':

case '-':

return 1;

case '\*':

case '/':

return 2;

case '^':

return 3;

}

return -1;

}

int main(){

init();

char \* s;

s=(char\*)malloc(100000\*sizeof(char));

scanf("%[^\n]s",s);

char \* ans;

ans=(char\*)calloc(10000,sizeof(char));

for(int i=0;i<strlen(s);i++){

if(s[i]==' '){

continue;

}

char c=s[i];

if(c-'0'>=0 && c-'0'<=9 || c>='a' && c<='z' || c>='A' && c<='Z'){

strncat(ans,&c,1);

}else if(c=='('){

push(c);

}else if (c==')')

{

while (isEmpty()!=1 && peek()!='(')

{

char d=peek();

strncat(ans,&d,1);

pop();

}

pop();

}else

{

while (isEmpty()!=1 && prec(c)<=prec(peek()))

{

char d=peek();

strncat(ans,&d,1);

pop();

}

push(c);

}

}

while (isEmpty()!=1)

{

if(peek()=='('){

printf("Invalid Input");

}

char d=peek();

strncat(ans,&d,1);

pop();

}

printf("%s",ans);

return 0;

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

Conclusion:

The proposed algorithm has a runtime of O(n), where n is the length of the input string .

Limitations and assumptions for this algorithm include:

1. Here length of the input string always have to be lesser or equal to 100000 & the stack also cannot contains more elements than 100000.
2. The operands used in the string should be either 0 to 9 or ‘a’ to ‘z’ or ‘A’ to ‘Z’ and the operator ‘+’,’-‘,’/’,’\*’,’^’.