23/CS/345

Rishu Singh

Page No.
Date

Data Structures

Algorithm

1) ~~Iterate~~ Iterate all the element of ~~string~~ character array (String is not present in C as a data type)

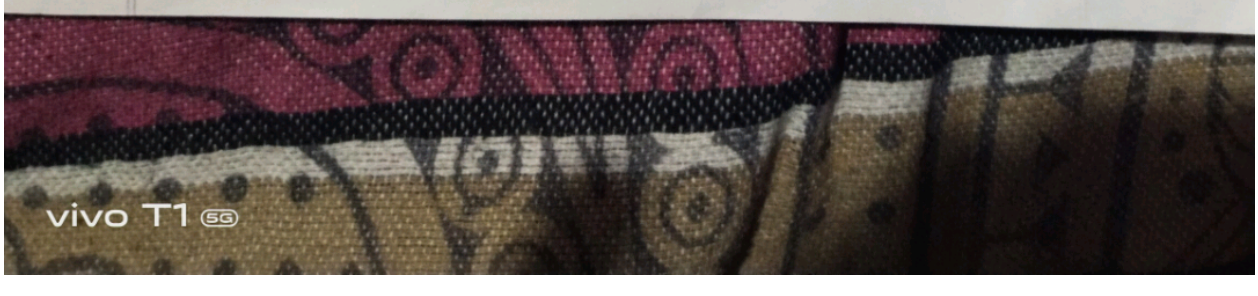
2) If current symbol is operand then append it to the postfix expression (another char array).

3) If current element is a bracket ('(') then simply push it in stack.
open

4) if current element is closed bracket (')') then pop the stack and put all popped elements in postfix expression /array until ')' is ~~encounted~~ encountered. Then pop ('(') also.

5) If current element is a operator (+, -, *, /, ^) then check if stack is empty push the operator in stack and if stack is not empty compare precedence of current operator with operator at stack top. If precedence of current operator is more, then push the operator in stack, if not then pop the stack, put/append popped elements in the postfix expression, If precedence is same, still do the pop And append popped operator in postfix expression.

6) Once the ~~int~~ iteration is completed (s[i] = '\0') then pop the stack until its empty then append the popped operator at postfix expression end And ~~return the~~ output the expression.



Page No.
Date

Implementation

Pseudocode

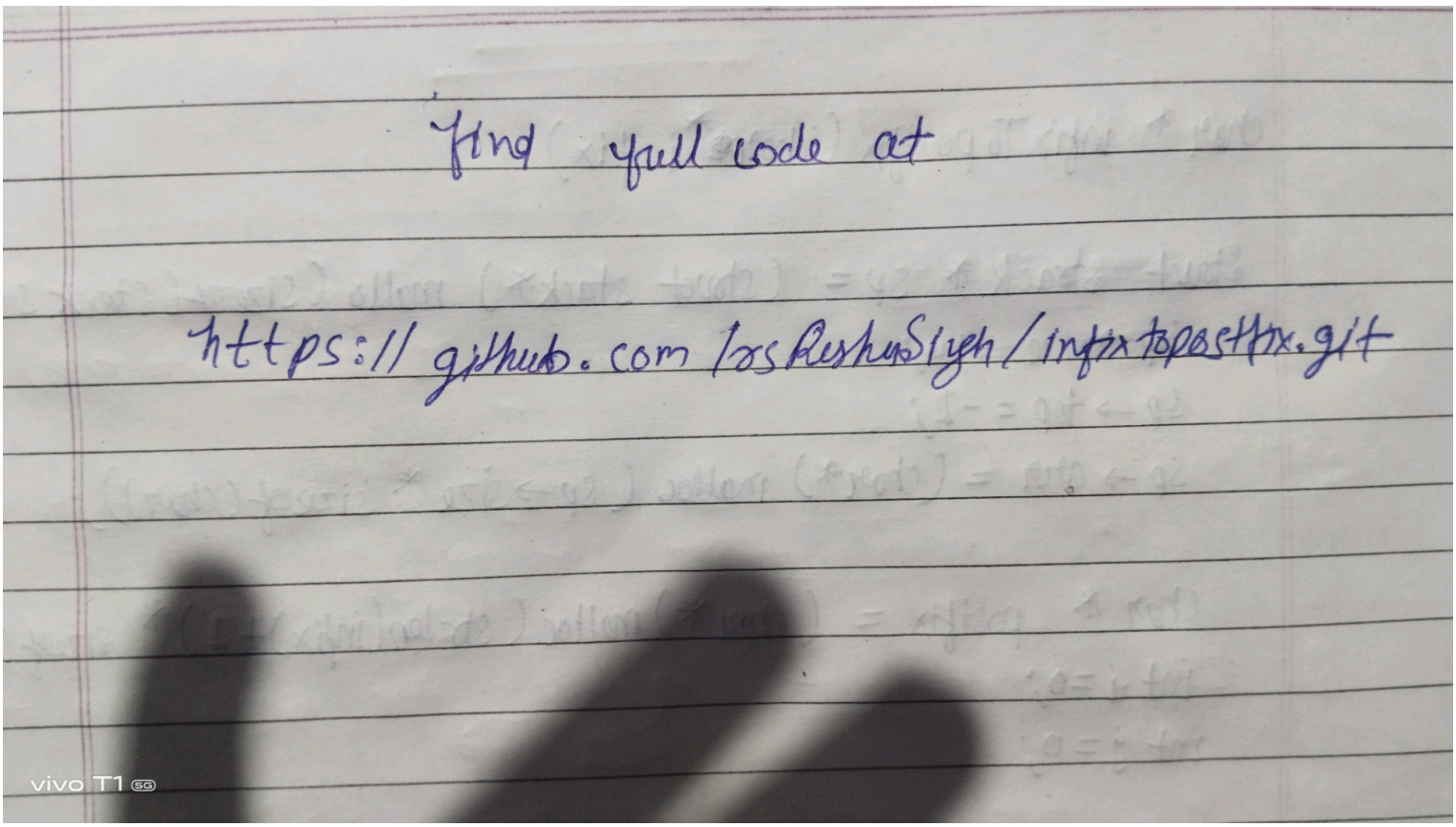
in stack_imp.c

Assumption : All the stack functions are already implemented
: Expression will only contain operands, '(', ')', "+/-*^"

```
#include <stdio.h>
#include <string.h>
#include "stack_imp.c"        // contains stack implementation

int precedence (char ch)
{
  if (ch == '*' || ch == '/')
      return 3;
  else if (ch == '^')
      return 4;
  else if (ch == '+' || ch == '-')
      return 2;
  else
  { printf("operator not supported \n");
    return -1;
  }
}
int isOperator (char ch) {
  if ( ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch == '^')
   { return 1; }
  else
      return 0;
}
```

Page No.
Date

```
char * infixToPostfix (char* infix)

struct stack * sp = (struct stack*) malloc (sizeof(struct stack));
  sp->size = 10;
  sp->top = -1;
  sp->arr = (char*) malloc (sp->size * sizeof(char));

  char * postfix = (char*) malloc (strlen(infix)+1)* sizeof(char);
  int i=0;
  int j=0;

  while( infix[i]!='\0')
{
  if(!isoperator(infix[i])){
      postfix[j++] = infix[i++];
  }
  else if( infix[i]=='(')

  else if( infix[i]==')' )
   {          while(      !='(')


   }
  else
```


find full code at
https://github.com/rsRishuSingh/infixtopostfix.git