DSA LAB TASK

SANJIDUR RAHMAN 2307006

Contents

1 DSA 1
1.1 main [166 lines] - 7ccbf513 1

1 DSA

1.1 main [166 lines] - 7ccbf513

```cpp
#include <bits/stdc++.h>
using namespace std;
#ifndef ONLINE_JUDGE
#include "/home/prantor/Coding/CP/debug.hpp"
#else
#define debug(x)
#define error(...)
#endif
#define all(x) (x).begin(), (x).end()
template <class T> class Stack {
public:
  T *arr;
  Stack(int size) { arr = new T[size]; }
  int idx = -1;
  void push(T val) { arr[++idx] = val; }
  void pop() { idx--; }
  bool empty() { return idx == -1; }
  void print() {
    for (int i = 0; i < idx; i++)
      cerr << arr[i] << " ";
    cerr << endl;
  }
  T top() { return arr[idx]; }
};
bool operand(char c) { return (c >= '0' and c <= '9'); }
int pre(char c) {
  if (c == '^')
    return 3;
  else if (c == '*' or c == '/')
    return 2;
  else if (c == '+' or c == '-')
    return 1;
  return -1;
}
string postfix(string s) {
  string ans = "";
  int n = s.size();
  Stack<char> st(n);
  for (int i = 0; i < n; i++) {
    char c = s[i];
    if (operand(c))
      ans += c;
    else if (c == '(')
      st.push(c);
    else if (c == ')') {
      while (st.top() != '(') {
        ans += st.top();
        st.pop();
      }
      st.pop();
    } else if (pre(c) != -1) {
      while (!st.empty() and pre(st.top()) >= pre(c)) {
        ans += st.top();
        st.pop();
      }
      st.push(c);
    }
  }
  while (!st.empty()) {
    ans += st.top();
    st.pop();
  }
  return ans;
}
string prefix(string s) {
  reverse(s.begin(), s.end());
  int n = s.size();
  for (int i = 0; i < n; i++) {
    if (s[i] == '(')
      s[i] = ')';
    else if (s[i] == ')')
      s[i] = '(';
  }
  string ans = postfix(s);
  reverse(ans.begin(), ans.end());
  return ans;
}
int f(char c, int a, int b) {
  if (c == '^')
    return a ^ b;
  if (c == '*')
    return a * b;
  if (c == '/')
    return a / b;
  if (c == '+')
    return a + b;
  if (c == '-')
    return a - b;
  return 0;
}
int eval_prefix(string s) {
  int n = s.size();
  Stack<int> st(n);
  for (int i = n - 1; i >= 0; i--) {
    if (operand(s[i])) {
      st.push(s[i] - '0');
    } else {
      int op1 = st.top();
      st.pop();
      int op2 = st.top();
      st.pop();
      st.push(f(s[i], op1, op2));
    }
  }
  return st.top();
}
int eval_postfix(string s) {
  s = postfix(s);
  int n = s.size();
  Stack<int> st(n);
  for (int i = 0; i < n; i++) {
    if (operand(s[i]))
      st.push(s[i] - '0');
    else {
      int op2 = st.top();
      st.pop();
      int op1 = st.top();
      st.pop();
      st.push(f(s[i], op1, op2));
    }
  }
  return st.top();
}
int main() {
  string s;
  getline(cin, s);
  int cnt = 0;
  for (auto i : s)
    if (i == '}')
      cnt++;
  cout << "Total Expressions Found: " << cnt << endl;
  Stack<string> expressions(cnt);
  Stack<string> prefixExpressions(cnt);
  Stack<int> results(cnt);
  int n = s.size();
  for (int i = 0; i < n; i++) {
    if (s[i] == '{') {
      string expr = "";
      i++;
      while (i < n && s[i] != '}') {
        expr += s[i];
        i++;
      }
      expressions.push(expr);
    }
  }
  while (!expressions.empty()) {
    string pre_expr = prefix(expressions.top());
    int result = eval_prefix(pre_expr);
    prefixExpressions.push(pre_expr);
    results.push(result);
    expressions.pop();
  }
  int counter = 0;
  cout << "Corresponding Prefix Expressions:" << endl;
  while (!prefixExpressions.empty()) {
    cout << ++counter << ". " << prefixExpressions.top()
         << endl;
    prefixExpressions.pop();
  }
  counter = 0;
  cout << "Result:" << endl;
  while (!results.empty()) {
    cout << ++counter << ". " << results.top() << endl;
    results.pop();
  }
}
```