McMaster University
MECHTRON 2MD3: Data Structures and Algorithms for Mechatronics

Winter 2023
Assignment 4
Due: March 16, 2023 at 12:30 pm

Instructions
Your solution should be submitted as a single C++ source file named "123456-asg4.cpp" where 123456 is your student ID.

Question [programming, 20 marks]
Extend the LinkedBinaryTree class from Goodrich text chapter 7 to represent expression trees. To test your class, you must use the provided main method that takes, as input, a list of postfix arithmetic expressions, converts them each into a binary expression tree, and stores each tree in a vector of trees.

Your trees must support the following operations: + - * / abs >
where "abs" returns the absolute value of its argument and ">" returns 1 if its first argument is greater than its second, and –1 otherwise.
Allow for the leaves of your expression tree to store variables a or b which are initially 0.

After creating the expression trees, your program will read values for variables a and b from a file, execute each expression tree for each <a, b> pair, and store the average result as that tree's "score". You program must work with the main method on page 2. Input and output examples are on page 3.

To accomplish this task, you will need to extend the LinkedBinaryTree class to include *at least* the following methods:

1. `void printExpression()`
   Recursively print an expression tree with parentheses. For example, postfix expression "1 2 + 3 *" is printed as "((1+2)*3)"

2. `double evaluateExpression(double a, double b)`
   `Recursively` evaluate an expression given values for variables a and b. Return the result.

3. `double getScore()`
   Return this tree's score.

4. `void setScore(double s)`
   `Set this tree's score.`

5. Overload the "<" operator to compare two LinkedBinaryTrees by their score.

Helper Function:
`LinkedBinaryTree createExpressionTree(string postfix)`
Builds and returns a binary expression tree from a string expression in postfix notation.

```
int main() {
  // Read postfix expressions into vector
  vector<LinkedBinaryTree> trees;
  ifstream exp_file("expressions.txt");
  string line;
  while (getline(exp_file, line)) {
    trees.push_back(createExpressionTree(line));
  }

  // Read input data into 2D vector
  vector<vector<double>> inputs;
  ifstream input_file("input.txt");
  while (getline(input_file, line)) {
    vector<double> ab_input;
    stringstream ss(line);
    string str;
    while (getline(ss, str, ' ')) {
      ab_input.push_back(stod(str));
    }
    inputs.push_back(ab_input);
  }

  // Evaluate expressions on input data
  for (auto& t : trees) {
    double sum = 0;
    for (auto& i : inputs) {
      sum += t.evaluateExpression(i[0], i[1]);
    }
    t.setScore(sum/inputs.size());
  }

  // Sort trees by their score
  sort(trees.begin(), trees.end());

  for (auto& t : trees) {
    cout << "Exp ";
t.printExpression();
cout << " Score " << t.getScore() << endl;
}
}
```

Expressions file: gitlab 2md3_2023/assignment_4_data/expressions.txt

```
-3.7 abs
a b >
3.7 -1.2 * abs
1.2 1.2 + 9 >
4 5 * 99.7 - 0.7 >
3.7 1.3 > 6 /
a b *
a 2.7 *
a b > abs 7 /
```

Input file: gitlab 2md3_2023/assignment_4_data/input.txt

```
7 7
77 -77
7.7 0.7
0.7 7
3.7 44
1.5 0.3
```

Expected output:

```
Exp (a * b) Score -951.077
Exp ((1.2 + 1.2) > 9) Score -1
Exp (((4 * 5) - 99.7) > 0.7) Score -1
Exp (a > b) Score 0
Exp (abs((a > b)) / 7) Score 0.142857
Exp ((3.7 > 1.3) / 6) Score 0.166667
Exp abs(-3.7) Score 3.7
Exp abs((3.7 * -1.2)) Score 4.44
Exp (a * 2.7) Score 43.92
```