COSC2430 GA1: Recursion, Stack, Linked List

Introduction

Scarlet and Travis are twin siblings who got separated at birth. Each of them was given a piece of dissembled jewelry with encodings in the form of arithmetic expressions. After reuniting with each other, Scarlet and Travis think the encodings on the jewelry hold the secret passcode to unlock a treasure.

Your task is to write a program to help the twins decode the numbers on both jewelry pieces and put them together to find the secret passcode.

Assumptions:

- Each input file may contain a maximum of 100 expressions.
- An input file may contain empty lines between expressions; you may ignore those lines.
- There will be no space in each expression.
- In each expression, only unit digits (0-9) will be used, and each operand will be a single digit.
- Operators: +,−.
- Parentheses: (,), {, }, [,].

Rules and Operations

- You will get a list of arithmetic expressions along with other types of information in an input file. For arithmetic expressions, each line will be considered a single expression. You first need to check whether the expression is valid or not before proceeding to decode the expression. An expression will be invalid if opening and ending parentheses do not match.

- The pieces cannot be assembled unless each piece has been entirely decoded on its own. In other words, you can't insert ALL of the decoded numbers to only one linked list during the decoding stage. Instead, make two separate linked lists for both twins to store the decoded numbers on each of their jewellery pieces, respectively.

- Validating and decoding the expressions:
    - The result of a valid expression gives you a number, which is either a part of the passcode or an instruction:

- ➢ During the decoding, add each **POSITIVE** number to the appropriate linked list (one linked list for each twin).
- ➢ In the case that the expression evaluates to a **NEGATIVE** number, find its positive counterpart in the list and remove that number from the list. If there is no positive number in the list, search for the number in the other list since the jewellery was broken into two pieces, and so was the list of encoded numbers.

- To obtain the final passcode:
  - Merge the two linked lists and sort in ascending order.
  - Count of total **invalid** expressions gives you an instruction that tells you to delete an element at the location with that number as the index. For example, if there are 5 invalid expressions, you will need to delete the element at index 5 (6th element) in the sorted linked list.

Note: All linked list operations must be implemented using recursion!!!

Examples

Example 1:

Input11.txt

Scarlet
3-4-5+1+7 // positive (2)
// Empty line
2+{[(6-9)+9]-4} // positive (4)
// Empty line
Travis
0-8-1-2+9 // negative (-2)
{4-[(4-9-0)]-2} // positive (7)
Scarlet
8-{{(1-[0+4]+8)}} //positive (3)
1-(4+5-7)+2 // positive (1)
9-{(1-2-3)+5} // positive (8)
Passcode:13478

Output11.txt

Scarlet: [4,3,1,8]

Travis: [7]

Decoded passcode: | 1 | 3 | 4 | 7 | 8 |

Actual passcode: | 1 | 3 | 4 | 7 | 8 |

Treasure unlocked!

Command line:

./decode “input=input11.txt;output=output11.txt” **or** ./decode input=input11.txt output=output11.txt

Example 2:

Input12.txt

Scarlet

// Empty line

// Empty line

9-{(1-2-3)+5} // positive (8)

9-2-3-4+1 // positive (1)

1-{(3-0-2)-6+3-{(1-4-6)-9 // invalid

{1-[(3-3+3)]+3} // positive (1)

Travis

1+{[[[(2+5)+1]]]-6} // positive (3)

{4-[(4-9-0)]-2} // positive (7)

1-[(5+2-9)+9] // negative (-6)

// Empty line

9-{(1-2-3)+5} // positive (8)

{1+[(2+5)+1]-6} // positive (3)

// Empty line

Scarlet

{{9-{(1-2-3)+4}}} // positive (9)

{4+[(4+9-0)]-8-3} // positive (6)

Travis

8-{(5+0-1))-3 // invalid

Passcode:1137889

Output12.txt

Scarlet: [8, 1, 1, 9]

Travis: [3, 7, 8, 3]

Decoded passcode: | 1 | 1 | 3 | 7 | 8 | 8 | 9 |

Actual passcode: | 1 | 1 | 3 | 7 | 8 | 8 | 9 |

Treasure unlocked!

Command line:

./decode “input=input12.txt;output=output12.txt” **or** ./decode input=input12.txt output=output12.txt

Example 3:

Input13.txt

Scarlet

// Empty line

8-{(7+0-1))-3 // invalid

// Empty line

1+{(5-[8-4)}-9] // invalid

{4+[(4+9-0)]-8-4} // positive (5)

9-{(4-2-3)+5} // positive (5)

{{2+{[(6-9)+9]-4}}} // positive (4)

Travis

3-4-5+1+7 // positive (2)

[1-(4+5-7)+3] // positive (2)

1+{[[[(2+5)+1]]]-7} // positive (2)

// Empty line

// Empty line

Scarlet

[1-(4+5-7)+3] // positive (2)

[0+{[(0-8-1-2)]}+9] // negative (-2)

0-8-1-2+9 // negative (-2)

Travis

0-8-1-2+9 // negative (-2)

{4+[(4+9-0)]-8-3} // positive (6)

Scarlet

{[2+[(4+9-0+2)]-8]-3} // positive (6)

// Empty line

Travis

[1-(4+2+9)+8] // negative (-6)

7-{(1+2-4)-0 // invalid

Passcode:2456

Output13.txt

Scarlet: [5, 5, 4, 6]

Travis: [2]

Decoded passcode: | 2 | 4 | 5 | 6 |

Actual passcode: | 2 | 4 | 5 | 6 |

Treasure unlocked!

Command line:

./decode “input=input13.txt;output=output13.txt” **or** ./decode input=input13.txt output=output13.txt

Important direction to submit your GA1:

We expect every student of each group will participate to solve the problem and discuss with each other. ***We will count the lowest grade of a group for every group member***, so make sure everyone in your team is submitting the same copy. The purpose of this GA is to learn the basic functionalities of Stack and Linked List. If someone has trouble understanding Stack or Linked List, they should discuss with their friends to get a clear understanding. You are encouraged to help your friends, but do not copy paste from other groups. If you help, it will also help you gain a deeper understanding of the topics.

Every student of each group needs to submit ***the same copy*** of the solution. GA1 needs to be turned in to our Linux server. Make sure to create a folder under your root directory, name it ***“ga1”*** (name must be in lower case and exact), only copy your .cpp and .h files to this folder, no test case or other files needed. If you use ArgumentManager.h, don’t forget to turn it in, too. ***GAs will be graded only once.*** You will not get the chance of resubmission after grading. So, make sure you are submitting the correct solution before the due date.