CENG114 MINI PROJECT

For the mini-project, students will form project groups (3-5 students team). You will submit the project (only one student in the team will submit it). Put a readme.txt file in your zip file so that I can see group members and any messages you want to deliver about your project submission. In zip file there will be only .java files and readme.txt file.I will examine it then I may ask you to present it to me. If there is a presentation, then I will ask questions to you so it is not guaranteed that all project members will take the same grades from the project.

You may use only the techniques and libraries we covered in the course.

In this mini project, you will try to obtain a Value (or closest Value) for given Values. Note that this value can be any kind of object, i.e. numbers. Operators are addition, subtraction, and multiplication and their behavior on values is defined by you. As you can see in this problem order of operator is not important since parenthesis are used.

For example, if the values are in number type and input numbers are 11, 7, 12, 6 and output number is 41 then you should output

```
(((12 - 7) * 6) + 11) = 41
```

since

```
(((12 - 7) * 6) + 11)      >      ((5 * 6) + 11)      >      (30 + 11)      >    41
```

Please also note that, you do not need to use all values to reach output value.

As another example, your inputs are characters (single or multiple) such as R, A, N, A and desired output is NAR then you should output

```
(((N + R) * A) - A)    >    ((NR * A)  - A)    >    (NARA - A)    >    NAR
```

As you see you also need to change the behavior of addition, subtraction, and multiplication for letters. Distance between the obtained word (NAR) and target word (NAR) is zero so you did it. But distance between NAR and NBR is -1 (see the shared distance function for finding the distances between 2 strings).

So here, + concatenates characters/strings, * puts the second one after each letter of the first one, and - removes the character from the first one.

```
Example 1: KABA - C   >  KABA   (C is not found)
Example 2: KABA - B   >   KAA
Example 3: KABA - A   >   KAB
```

- operators are fixed: addition, subtraction, multiplication

- output is single but input variables can be in any number. so there can be 3 inputs, or 5, or 30. so your implementation should able to get any number of input. you do not need to get input and output data from console. just create them in main method as your test cases, then I can change these lines of code to test my own inputs. I urge you to use recursion in this mini project.

- input variable can be any type. it can be integer, it can be character, it can be double, or it can be an enumeration that you defined yourself. For example: Enum Animals {Cat, Dog, Cow, Snake};

so your input type must be an abstract class and you should derive your input data type classes from that.

- addition, subtraction, multiplication behavior should be abstract and you should define your sub classes such that you will addition, subtraction, multiplication for integer, and you should have another addition, subtraction, multiplication for letters etc. for example Cat + Dog can be Cow if you defined addition in that behavior.

Use is-a and has-a mechanisms, inheritance, interfaces, aggregation, composition, polymorphism, constructors etc. and create a good and flexible (and extendable) software.