CSE142—Computer Programming I
Programming Assignment #7
due: Tuesday, 5/24/22, 11 pm

This assignment will give you practice with arrays and producing an external output file. You are going to write a program that processes an input file of data for a personality test known as the Keirsey Temperament Sorter. The Keirsey personality test involves answering 70 questions each of which have two answers. We will refer to them as the "A" answer and the "B" answer. People taking the test are allowed to leave a question blank, in which case their answer will be recorded with a dash ("-").

The input file will contain a series of line pairs, one per person. The first line will have the person's name (possibly including spaces) and the second line will have a series of 70 letters all in a row (all either "A", "B" or "-"). Your job is to compute the scores and overall result for each person and to report this information to an output file.

The Keirsey test measures four independent dimensions of personality:

> Extrovert versus Introvert (E vs I): what energizes you
> Sensation versus iNtuition (S vs N): what you focus on
> Thinking versus Feeling (T vs F): how you interpret what you focus on
> Judging versus Perceiving (J vs P): how you approach life

Individuals are categorized as being on one side or the other of each of these dimensions. The corresponding letters are put together to form a personality type. For example, if you are an extravert, intuitive, thinking, perceiving person then you are referred to as an ENTP. Usually the letter used is the first letter of the corresponding word, but notice that because the letter "I" is used for "Introvert", the letter "N" is used for "iNtuition."

Remember that the Keirsey test involves 70 questions answered either A or B. The A answers correspond to extravert, sensation, thinking and judging (the left-hand answers in the list above). The B answers correspond to introvert, intuition, feeling and perceiving (the right-hand answers in the list above). For each of these dimensions, we determine a number between 0 and 100 and indicate whether they were closer to the A side or the B side. The number is computed by figuring out what percentage of B answers the user gave for that dimension (rounded to the nearest integer).

Let's look at a specific example. Suppose that someone's answers divide up as follows:

Dimension	# of A answers	# of B answers	% B	Result
Extrovert/Introvert	1	9	90%	I
Sensing/iNtuition	17	3	15%	S
Thinking/Feeling	18	2	10%	T
Judging/Perceiving	18	2	10%	J

These numbers correspond to the answers given by the first person in the sample input file ("Betty Boop"). We add up how many of each type of answer we got for each of the four dimensions. Then we compute the percentage of B answers for each dimension. Then we assign letters based on which side the person ends up on for each dimension. In the Extrovert/Introvert dimension, for

example, the person gave 9 “B” answers out of 10 total, which is 90%, which means they end up on the B side which is “Introvert” or I. In the Sensing/iNtuition dimension the person gave 3 “B” answers out of 20 total, which is 15%, which means they end up on the A side with is “Sensing” or S. The overall scores for this person are the percentages (90, 15, 10, 10) which works out to a personality type of ISTJ.

Some people will end up with a percentage of 50 in one or more dimensions. This represents a tie, where the person doesn’t clearly fall on either side. In this case we use the letter “X” to indicate that the person is in the middle for this particular dimension. The last two entries in the sample input file end up with X’s in their personality type.

Take a moment to compare the sample input file and the sample output file and you will see that each pair of lines in the input file is turned into a single line of output in the output file that reports the person’s name, the list of percentages and the personality type. You are required to exactly reproduce the format of this output file. You are also required to reproduce the sample log of execution. You will provide a short introduction and then ask the user for the names of the input file and output file.

If you are interested in taking the personality test yourself, you will find a link from the class webpage to an online form with the 70 questions. We will be making a special data file called bigdata.txt that includes data from students in the class.

To count the number of A and B answers for each dimension, you need to know something about the structure of the test. You will get the best results if you take the test without knowing about the structure, so you might want to take the test first before you read what follows. The test has 10 groups of 7 questions with a repeating pattern in each group of 7 questions. The first question in each group is an Extrovert/Introvert question (questions 1, 8, 15, 22, etc). The next two questions are for Sensing/iNtuition (questions 2, 3, 9, 10, 16, 17, 23, 24, etc). The next two questions are for Thinking/Feeling (questions 4, 5, 11, 12, 18, 19, 25, 26, etc). And the final two questions in each group are for Judging/Perceiving (questions 6, 7, 13, 14, 20, 21, 27, 28, etc). Notice that there are half as many Extrovert/Introvert questions as there are for the other three dimensions. The seventy letters in the input file appear in question order (first letter for question 1, second letter for question 2, third letter for question 3, etc).

Remember that the user might leave a question blank, in which case you will find a dash in the input file for that question. Dash answers are not included in computing the percentages. For example, if for one of the dimensions you have 6 A answers, 9 B answers and 5 dashes, you would compute the percentage of B answers as 9 of 15, or 60%.

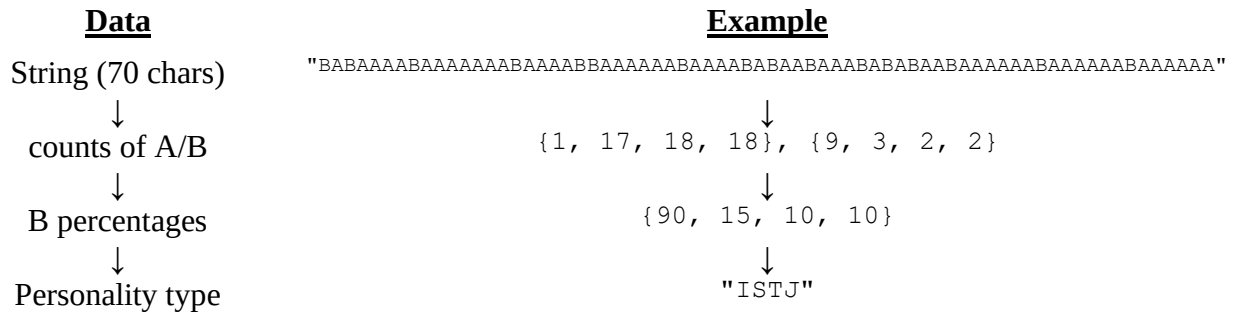
You should round percentages to the nearest integer. You can use the Math.round method to do so, but you will have to cast the result to an int, as in:

```
int percent = (int) Math.round(percentage);
```

This is the first time you will generate an output file. You do so by constructing an object of type PrintStream and writing to it in the same way you write to System.out (with print and println statements). See section 6.4 of the book for examples. It is a good idea to send your output to System.out while you are developing your program and send it to the output file only after you have thoroughly tested your program.

You should read the user’s answers from the console using a call on nextLine(). This will read an entire line of input and return it as a String.

One of the things to keep in mind for this program is that you are transforming data from one form to another. You start with a String that has 70 characters in it. You convert that into two sets of counters (how many A answers for each dimension, how many B answers for each dimension). You convert that into a set of percentages. And you finally convert that into a String that represents the personality type. If you work through this step by step, the problem will be easier to solve. The following diagram summarizes the different transformations:



Notice that the letters "A" and "B" in the sample input file sometimes appear as uppercase letters and sometimes appear as lowercase letters. Your program must recognize them in either case.

You may assume that the input file has no errors. In particular, you may assume that the file exists, that it is composed of pairs of lines, and that the second line in each pair will have exactly 70 characters that are either A, B or dash (although the A's and B's might be in either uppercase form or lowercase form or a combination). You may also assume that nobody has zero answers for a given dimension (it would be impossible to determine a percentage in that case).

The sample input and output files provide just a few simple examples of how this program works. We will be using a much more extensive file for testing your program. As mentioned earlier, we will include data from people in the class to make this file.

Your program is likely to have the number "4" in several places because of the four dimensions of this test. You should introduce a class constant to make this more readable instead of using 4 itself. It won't be possible, however, to change this constant to some other number and have the program function properly. The constant is helpful for documentation purposes, but it won't make the program particularly flexible.

We will once again be expecting you to use good programming style and to include useful comments throughout your program. We are not specifying how to decompose this problem into methods, but we will be grading on the quality of your decomposition. That means you will have to decide how to decompose the program into methods. You should keep in mind the ideas we have been stressing all quarter. You don't want to have redundant code. You don't want to have any one method be overly long. You want to break the problem down into logical subproblems so that someone reading your code can see the sequence of steps it is performing. You want main to be a concise summary of the program. You want to avoid method chaining by using parameters and return values. You should restrict yourself to the programming constructs included in chapters 1 through 7 of the textbook in solving this problem. You are restricted to one-dimensional arrays for this assignment (no 2-dimensional arrays, 3-dimensional arrays, etc). You are allowed to use Arrays.toString, but you should not use any other methods from the Arrays class for this assignment.

Your program should be stored in a file called Personality.java. You will need to include the file personality.txt in the same folder as your program.

You can find out more about the Keirsey Temperament Sorter at http://www.keirsey.com.

Input file personality.txt

```
Betty Boop
BABAAAABAAAAAAABAAAABBAAAAAABAAAABABAABAAABABABAABAAAAAABAAAAAABAAAAAA
Snoopy
AABBAABBBBBABABAAAAABABBAABBAAAABBBAAABAABAABABAAAABAABBBBAAABBAABABBB
Bugs Bunny
aabaabbabbbaaaabaaaabaaaaababbbaabaaaabaabbbbabaaaabaabaaaaaabbaaaaabb
Daffy Duck
BAAAAA-BAAAABABAAAAAABA-AAAABABAAAABAABAA-BAAABAABAAAAAABA-BAAABA-BAAA
The frumious bandersnatch
-BBaBAA-BBbBBABBBBA-BaBBBBBbbBBABBBBBBABB-BBBaBBABBBBBBB-BABBBBBBBBBBB
Minnie Mouse
BABA-AABABBBAABAABA-ABABAAAB-ABAAAAAA-AAAABAAABAAABAAAAAB-ABBAAAAAAAAA
Luke Skywalker
bbbaaabbbbaaba-BAAAABBABBAAABBAABAAB-AAAAABBBABAABABA-ABBBABBABAA-AAAA
Han Solo
BA-ABABBB-bbbaababaaaabbaaabbaaabbabABBAAABABBAAABABAAAABBABAAABBABAAB
Princess Leia
BABBAAABBBBAAABBA-AAAABABBABBABBAAABAABAAABBBA-AABAABAAAABAAAAABABBBAA
```

Output file for personality.txt

```
Betty Boop: [90, 15, 10, 10] = ISTJ
Snoopy: [30, 45, 30, 70] = ESTP
Bugs Bunny: [20, 45, 15, 55] = ESTP
Daffy Duck: [100, 6, 20, 6] = ISTJ
The frumious bandersnatch: [86, 95, 75, 78] = INFP
Minnie Mouse: [67, 28, 32, 5] = ISTJ
Luke Skywalker: [89, 61, 26, 25] = INTJ
Han Solo: [80, 50, 45, 25] = IXTJ
Princess Leia: [80, 50, 50, 5] = IXXJ
```

Sample log (user input bold and underlined)

```
This program processes a file of answers to the
Keirsey Temperament Sorter.  It converts the
various A and B answers for each person into
a sequence of B-percentages and then into a
four-letter personality type.

input file name? personality.txt
output file name? output.txt
```