COMPUTER SCIENCE 12B (SPRING 2020)
PROGRAMMING IN JAVA

PROGRAMMING ASSIGNMENT 1

Program Description:

This assignment will test your understanding of the use of the `Scanner` objects, `printf`(), `Strings`, `for` loops, `if/else` statements , parameters and methods that return values.
Modularity in your code is very important, YOU MUST USE STATIC METHODS.

Write five programs to solve the following problems. Write each program in a different file with extension **.java** Named your programs/files **Problem1.java**, **Problem2.java**, etc.

Problem 1: Write a program that accepts the name (the name should not contain spaces), quantity, and price of three items. Output a bill with a tax rate of 6.25%. All prices should be displayed with two decimal places. The bill should be formatted in columns with 30 characters for the name, 10 characters for the quantity, 10 characters for the price, and 10 characters for the total. Sample input and output is shown as follows (user input underlined):

```
Input name of item 1:
lollipops
Input quantity of item 1:
10
Input price of item 1:
0.50
Input name of item 2:
diet_soda
Input quantity of item 2:
3
Input price of item 2:
1.25
Input name of item 3:
chocolate_bar
Input quantity of item 3:
20
Input price of item 3:
0.75

Your bill:
Item                    Quantity          Price             Total
lollipops               10                0.50              5.00
diet_soda               3                 1.25              3.75
chocolate_bar           20                0.75              15.00

Subtotal                                                    23.75
6.25% sales tax                                              1.48
Total                                                       25.23
```

Problem 2: Write a program that accepts two string variables, `first` and `last`, which the user should populate with his or her name. First, convert both strings to all lowercase. Your program should then create a new string that contains the full name in pig latin with the first letter capitalized for the first and last name. Use only the pig latin rule of moving the first letter to the end of the word and adding "ay". Output the pig latin name to the screen.

For example, if the user inputs "Antonella" for the first name and "Dilillo" for the last name, then the program should create a new string with the text "Ntonellaaay Ililloday" and print it.

Problem 3: Write a program to find a baseball player's batting average. The program should ask the user to enter the number of times the player was at bat and the number of hits earned. It should then display the batting average to four decimal places.

Problem 4: Write a program that produces a Caesar cipher of a given message string. A Caesar cipher is formed by rotating each letter of a message by a given amount. For example, if your rotate by 3, every A becomes D; every B becomes E; and so on. Toward the end of the alphabet, you wrap around: X becomes A; Y becomes B; and Z becomes C. Your program should prompt for a message and an amount by which to rotate each letter and should output the encoded message.

Log of execution (user input underlined)

```
Your message? Attack zerg at dawn
Encoding key? 3
Your message: DWWDFN CHUJ DW GDZQ
```

Problem 5: This program uses a student's grades on homework, a midterm exam, and a final exam to compute an overall course grade. The course grade is a weighted average. To compute a weighted average, the student's point scores in each category are divided by the total points for that category, then multiplied by that category's weight. (The sum of all categories' weights should be 100.)

$$Grade = WeightedHomeworkScore + WeightedMidtermScore + WeightedFinalExamScore$$

$$Grade = \left(\frac{HomeworkEarned}{HomeworkPossible} \times HomeworkWeight\right) + \left(\frac{MidtermEarned}{MidtermPossible} \times MidtermWeight\right) + \left(\frac{FinalEarned}{FinalPossible} \times FinalWeight\right)$$

Example: A course has 50% weight for homework, 20% weight for its midterm, and 30% weight for its final. There are 3 homework assignments worth 15, 20, and 25 points respectively. The student received homework scores of 10, 16, and 19, a midterm score of 81, and a final exam score of 73 (which was curved by +5 points to make a curved score of 78).

$$Grade = WeightedHomeworkScore + WeightedMidtermScore + WeightedFinalExamScore$$

$$Grade = \left(\frac{HomeworkEarned}{HomeworkPossible} \times HomeworkWeight\right) + \left(\frac{MidtermEarned}{MidtermPossible} \times MidtermWeight\right) + \left(\frac{FinalEarned}{FinalPossible} \times FinalWeight\right)$$

$$Grade = \left(\frac{10+16+19}{15+20+25} \times 50\right) + \left(\frac{81}{100} \times 20\right) + \left(\frac{73+5}{100} \times 30\right)$$

$$Grade = 37.5 + 16.2 + 23.4$$

$$Grade = 77.1$$

Note that the above math is written as real math and not Java math; in Java, an integer expression such as 81/100 would evaluate to 0, but above the value intended is 0.81.

This program asks the user to enter his/her grades on homework, a midterm exam, and (optionally) a final exam. Using this information, the program can either compute the student's overall course grade (if the student has taken the final exam) or tell the student what score he/she needs to earn on the final exam to achieve a certain grade in the class.

Problem 5 Program Behavior: The following section describes the program's behavior in detail. You can also get a good idea of the program's behavior by looking at the logs of execution that will follow, but you should read this section completely to make sure you see all of the cases and behaviors your program should have.

- First, the program prints a header message describing itself.

- Second, the program asks the user for homework information. The user enters the weight of the homework assignments, which can be any number between 0 and 100. The user enters how many homework assignments were completed. For each of these assignments, the user enters his/her score along with the maximum possible score, separated by spaces. Your code should work for any number of assignments greater than or equal to 1.

- Third, the program asks the user for midterm exam information. The user enters its weight from 0 to 100 and his/her score. (The user does not enter the maximum score for the midterm, because this is assumed to be 100.) The user is asked whether the midterm was curved; a response of 1 means yes, and 2 means no.

    - If there was a curve, the user is asked how many points were added. These points are added to the user's midterm score.

    - No exam's score can be above 100, so if the curve would have made the user's score exceed 100, a score of 100 is used.

- Fourth, the program asks the user for final exam information. The program can be run before or after the final exam, so first the user enters whether the final exam has been completed; a response of 1 means yes, and 2 means no.

    - If the final exam has been completed, the program will help the user show his/her overall course grade. The user enters the exam weight and his/her score on the final exam. The user is asked whether the final exam was curved; a response of 1 means yes, and 2 means no. If there was a curve, the user is asked how many points were added. These points are added to the user's final exam score. No exam's score can be above 100, so if the curve would have made the user's score exceed 100, a score of 100 is used. Lastly, the user's overall course grade is printed.

    - If the final exam has not yet been completed, the program will help the user see what final exam score he/she needs to earn a particular overall grade in the course. The final exam weight and the user's desired course grade are entered. The program computes and shows what score the student needs on the final to achieve the desired course grade.

- If the student can achieve the desired course grade without taking the final exam (in other words, if a final exam score of 0 or less is required), the program shows 0.0 as the needed final exam score.

- If the required score for the student to get the desired course grade is greater than 100, it should still be printed as normal, and then the user should receive a message indicating that the desired score cannot be achieved, and a message showing the highest course grade that the student can get (which is the grade that would result if the student earns 100 on the final exam).

See the provided logs of execution for an example of expected output.

Problem 5 Expected Output: The following logs of execution indicate the exact format of the output that you should reproduce. Your program's output should match these samples exactly when the same input is typed. Please note that there are some blank lines between sections of output and that some lines of output are indented by four spaces. Also note that input values typed by the user appear on the same line as the corresponding prompt message.

First log of execution (user input underlined)

```
This program accepts your homework and exam
scores as input, and computes your grade in
the course or indicates what grade you need
to earn on the final exam.

Homework:
    What is its weight (0-100)? 50
    How many homework assignments were there? 3
    Homework 1 score and max score: 14 15
    Homework 2 score and max score: 18 20
    Homework 3 score and max score: 19 25
    Weighted homework score: 42.5

Midterm exam:
    What is its weight (0-100)? 20
    Exam score: 81
    Was there a curve? (1 for yes, 2 for no) 2
    Weighted exam score: 16.2

Final exam:
    Have you taken the final exam yet? (1 for yes, 2 for no) 2
    What is its weight (0-100)? 30
    What percentage would you like to earn in the course? 80

    You need a score of 71.0 on the final exam.
```

Second log of execution (user input underlined)

```
This program accepts your homework and exam
scores as input, and computes your grade in
the course or indicates what grade you need
to earn on the final exam.

Homework:
    What is its weight (0-100)? 40
    How many homework assignments were there? 4
    Homework 1 score and max score: 21 30
```

```
    Homework 2 score and max score: 11 20
    Homework 3 score and max score: 28 50
    Homework 4 score and max score: 5 10
    Weighted homework score: 23.64

Midterm exam:
    What is its weight (0-100)? 30
    Exam score: 95
    Was there a curve? (1 for yes, 2 for no) 1
    How much was the curve? 10
    Weighted exam score: 30.0

Final exam:
    Have you taken the final exam yet? (1 for yes, 2 for no) 1
    What is its weight (0-100)? 30
    Exam score: 63
    Was there a curve? (1 for yes, 2 for no) 1
    How much was the curve? 5
    Weighted exam score: 20.4

    Your course grade is 74.04
```

Third log of execution (user input underlined)

```
This program accepts your homework and exam
scores as input, and computes your grade in
the course or indicates what grade you need
to earn on the final exam.

Homework:
    What is its weight (0-100)? 50
    How many homework assignments were there? 2
    Homework 1 score and max score: 10 50
    Homework 2 score and max score: 12 25
    Weighted homework score: 14.67

Midterm exam:
    What is its weight (0-100)? 25
    Exam score: 56
    Was there a curve? (1 for yes, 2 for no) 2
    Weighted exam score: 14.0

Final exam:
    Have you taken the final exam yet? (1 for yes, 2 for no) 2
    What is its weight (0-100)? 25
    What percentage would you like to earn in the course? 90

    You need a score of 245.33 on the final exam.
    Sorry, it is impossible to achieve this percentage.
    The highest percentage you can get is 53.67.
```

Problem 5 Input and Output Details: You do not have to perform any error checking on user input. You may assume that the user types legal values of the proper type and in the appropriate range. For example, when prompted for a number, assume that the user does enter a number and not a String. When prompted for a number within an expected range, such as a weight or exam score between 0 and 100, the user will enter a valid number in that range and not a number outside the range. When prompted for the number of homework assignments, the user will enter a number no less than 1. The sum of the weights the user will enter will always be 100. Notice that all real

numbers output by the program are printed with no more than 2 digits after the decimal point. Only round numbers as you are about to print them.

Guidelines:

Add a comment header for each program:

```
// FirstName LastName
// PA#1
// date
```

For this assignment you should limit yourself to the Java features covered in class so far (lecture 6). Although we will cover other topics while you are working on this assignment, do not use any of those features.

Grading:

You will be graded on

- **External Correctness:** The output of your program should match exactly what is expected. Programs that do not compile will not receive points for external correctness.
- **Internal Correctness:** Your source code should follow the stylistic guidelines shown in class. Also, remember to include the comment header at the beginning of your program.

Submission:

Create a folder containing your Java source code (programs). Compress (zip) the folder and upload it to Latte by the day it is due. For late policy check the syllabus.