ITSE 2417- Fall 2016
Program Set #2

See Java Grading /Program Guide Sheet for directions and grading/submission information.

1. One event in Athletics (Track and Field) at the 2016 Olympics was the heptathlon. The heptathlon is made up of 7 events (100m hurdles, high jump, shot put, 200m, long jump, javelin throw, and 800m). Write a Java program to calculate a competitor's total score in the heptathlon based on the seven events. The total score in heptathlon is computed by summing the scores in all 7 events. The events are split into three groups, and the scores are calculated according to the three formulas:

- Running events (200 m, 800 m and 100 m hurdles): $P = a * (b - T)^c$
- Jumping events (high jump and long jump): $P = a * (M - b)^c$
- Throwing events (shot put and javelin): $P = a * (D - b)^c$

`P` is for points, `T` is for time in seconds, `M` is for height or length in centimeters and `D` is length in meters. `a`, `b` and `c` have different values for each of the events see below.

Event	a	b	c	Type
100 m Hurdles	9.23076	26.7	1.835	Running
High Jump	1.84523	75	1.348	Field
Shot Put	56.0211	1.5	1.05	Field
200 m	4.99087	42.5	1.81	Running
Long Jump	0.188807	210	1.41	Field
Javelin Throw	15.9803	3.8	1.04	Field
800 m	0.11193	254	1.88	Running

The user will input seven values corresponding to the order of the events in the table. Output will be the competitor's total score for the heptathlon as an integer. Use a least one user-defined method in your program. Test your program with actual results from the 2016 Rio Olympics (provided). Output should be user friendly.

Note: https://en.wikipedia.org/wiki/Floor_and_ceiling_functions

Name the program: `HeptScoreXX.java`, where XX are your initials.

2. There are many words that contain all of the vowels, A, E, I, O, and U exactly once. There are fewer words that contain all of the vowels exactly once and the vowels appear in alphabetical order. For example, `SEQUOIA` is a word that contains all five vowels exactly once. `FACETIOUS` is a word contains all five vowels exactly once and the vowels appear in alphabetical order. `BUTTERFLY` is a word that does not contain all five vowels. Write a Java program that determines:
You are to write a program that will determine

- Whether all of the vowels appear in a given word exactly once.
- Whether all of the vowels appear in a given word exactly once and they appear in alphabetical order.
- Which vowels are missing if all of the vowels do not appear in the given word.

Assume all words to be tested will have at most one occurrence of any vowel. For each word inputted from the keyboard (convert to all caps), print the word and one of the following:

- `ALL, IN ORDER` if all five vowels appear in alphabetical order
- `ALL, NOT IN ORDER` if all five vowels appear but are not in alphabetical order, or
- `DOES NOT CONTAIN x y …` where x, y, etc. are the missing vowels, in alphabetical order and separated by a space, that do not appear in the word.

Use the `String` class and at least one user defined method. Output should look similar to below.

Sample Runs:

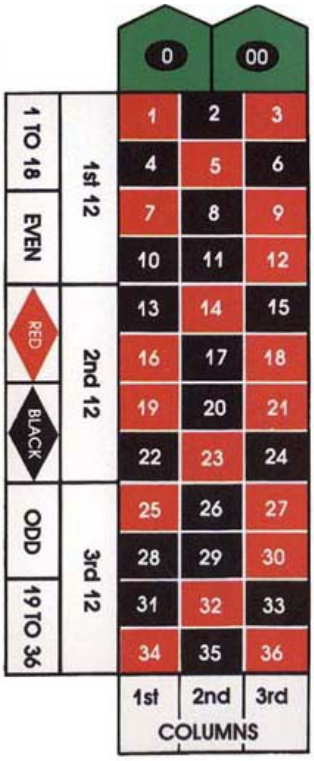
```
Enter a word: sequoia

SEQUOIA- ALL, NOT IN ORDER


Enter a word: FACETIOUS

FACETIOUS- ALL, IN ORDER


Enter a word: BUTTERFLY

BUTTERFLY- DOES NOT CONTAIN A I O
```

Name the program: `VowelsInWordsXX.java`, where XX are your initials.

3. Write a Java program find the inverse of a square matrix. Let the user input the matrix size (up to size 6) and randomly generate the elements in the matrix using values from -100 to 100. Error check input. Also, check to make sure if the matrix is singular. Using the `inversematrix` PDF file provided by your instructor, translate the C code to Java. Fix any errors with the code. Output should be user friendly.

Name the program: `MatrixInverseXX.java`, where XX are your initials.

4 (**). Write a Java program that simulates a roulette game. The roulette table has 36 numbers (1-36) that are arranged in three columns of 12 rows. The first row has the numbers 1 through 3; the second row contains 4 through 6, and so on. The numbers 0 & 00 are outside the table of numbers. The numbers in the table are colored red and black (0 & 00 are green). The red numbers are 1, 3, 5, 7, 9, 12, 14, 16, 18, 19, 21, 23, 25, 27, 30, 32, 34, and 36. The other half of the numbers are black.



In a simplified set of rules, players can bet on an individual number (including 0 & 00), Inside, and Outside bets (see below) The user should be allowed to enter one of the bets, and the application should use the `random()` method as the basis for computing the number that would be rolled on the wheel. It should then compare this number to the bet, and report whether it won or lost. Finally, the program terminates by prompting the user to quit the game.

Inside Bets

- Straight- single number bet
- Split- 2 adjacent numbers; vertical or horizontal (62 ways)
- Street-single horizontal line (row bet) (15 ways)
- Corner- block of four numbers (square) (22 ways)
- Line (Six Line)- two adjacent streets (two adjacent rows/six numbers) (11 ways)

Outside Bets

- Low (1-18)/High (19-36)-second 18 numbers
- Color (Red/Black)
- Even/Odd
- Column- any one of the three columns (vertical line)
- Dozen (1-12, 13-24, 25-36)- first, second or third group of 12 numbers

The application should also allow the user to enter an initial amount of money into an account. In addition to placing a bet, the user should be able to specify an amount to go with the bet; this amount should be deducted from the account. Any winnings should be added to the account according to the payout table below; for losses, only deduct (subtract) the bet amount stated above. The updated account value should then be printed out. The user should not be allowed to bet more than the total in the account, and only allowed to bet in 5 dollar increments. Winnings are computed as follows:

Bet Type	Payout
Straight Up	35 to 1
Split Bet	17 to 1
Street Bet	11 to 1
Corner Bet	8 to 1
Line Bet	5 to 1
Column Bet	2 to 1
Dozen Bet	2 to 1
Color Bet	1 to 1
Even/Odd Bet	1 to 1
Low/High Bet	1 to 1

Finally, implement a menu with two options representing different modes for establishing the number. The first option is the test mode- a test mode is required so the program can be graded. In test mode, the user enters both numbers (number and bet). This will allow you to test your logic, results, etc. The second option is game mode. In this mode, the program will randomly generate the number and the user can bet (guess) at the number. Have the user only once enter an initial choice of G or T mode at start of the program- the mode will remain until the user decides to quit. Output should look similar to below.

Sample Run:

```
Enter the playing mode: G

Enter starting money amount: 1000

Player starts with 1000 dollars

How much to bet? 50 dollars

Enter the type of bet:  1
1.    Single Number
2.    Adjacent Numbers
3.    Row
4.    Corner
5.    2 Rows
6.    Column
7.    Dozen
8.    Color
9.    Even/Odd
10.   Low/Hi

Enter the number to bet on:  30

The computer rolled: 14

You loose.

Player has 950 dollars.
```

```
Play again (Y/N)? y

Player has 950 dollars.

How much to bet? 400 dollars

Enter the type of bet:  6
1.    Single Number
2.    Adjacent Numbers
3.    Row
4.    Corner
5.    2 Rows
6.    Column
7.    Dozen
8.    Color
9.    Even/Odd
10.   Low/Hi

Enter the number to bet on:  23

The computer rolled: 2

You win!

Player has 1750 dollars. Play again (Y/N)? N
Thanks for playing
```

Name the program: `RouletteXX.java`, where XX are your initials.