**CSE 174 – Sections E-H – Fall 2016**

**PROGRAM #12: 60 points – Due Sunday, December 4, by 11:59 p.m.**

**Outcomes:**

* Write programs that use arrays
* Write programs that display numbers formatted according to a given specification
* Format and comment source code that adheres to a given set of formatting guidelines

**Scoring:**

At a bare minimum, the program you submit must have the assigned source code, and your source code must compile and run without crashing.

* If you submit source code that does not compile, your score will be zero.
* If you submit source code that compiles, but your instructor's tester does not compile,, your score will lose at least 50% of the credit. **It is very important to use the correct class name (YahtzeeGame), and all methods with the exact names, return types, parameters, and so on. Even if you do not know how to implement a method, write its stub because your code will be tested with a separate tester class. Failure to write methods according to requirements will result in at least a 50% deduction.**
* If you submit source code that roughly resembles the requirements and it compiles, but it crashes under normal operating conditions (nice input from the user), your score will be reduced by 75%.
* Deductions will be made for not meeting the usual formatting requirements (commenting, indentation, appropriate variable names, and so on.

|  |  |  |
| --- | --- | --- |
|  | **Full credit** | **No credit or Partial credit** |
| **Implement dice logic**  **(39 points)** | Program correctly determines the score for each type of roll, as well as the maximum score. | There are errors in the dice logic. |
| **Sort**  **(4 points)** | Program correctly uses Bubble Sort to sort the dice in ascending order. | There are errors in the sorting. |
| **Implement the reroll logic**  **(7 points)** | Program correctly handles the logic of rerolling the dice, including handling good and bad user input (the user will always enter a string with no spaces) | There are errors in the reroll logic. |
| **Implement overall game flow (7 points)** | Program works as shown in the sample runs | Program flow does not work as specified |
| **Write short methods**  **(3 points)** | Each method is at most 30 lines, including blank lines and comments. | Some methods exceed the 30-line limit. |

**Background:**

The intent of this exercise is to write a class (**YahtzeeGame**) that is a partial implementation of the game of Yahtzee (for more information on this game see Wikipedia). In this game, a player is presented with 5 dice. After throwing the dice, the player gets up to two opportunities to determine which of the 5 dice to throw again. The program displays the various scoring options to the user.

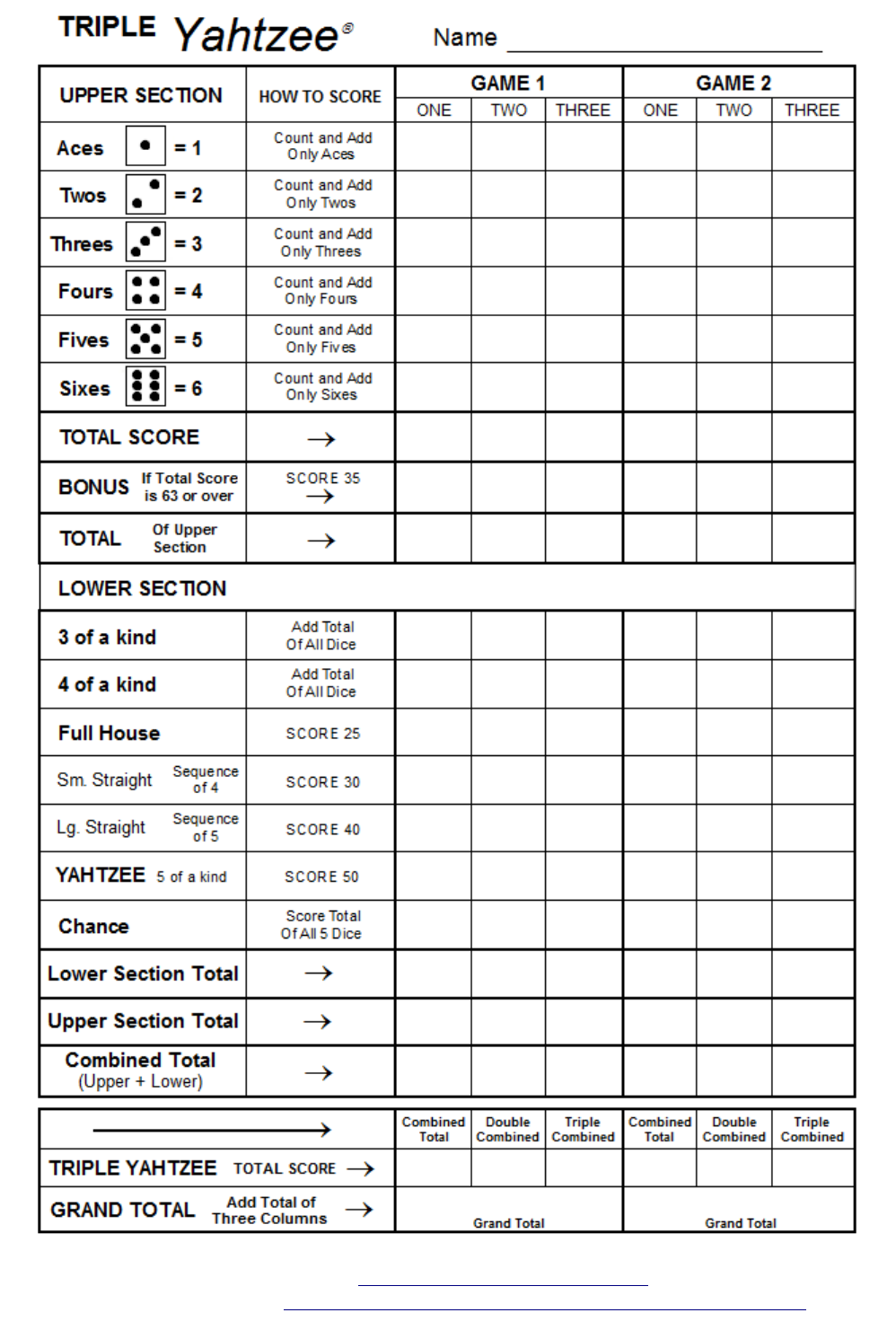
**Requirements:**

Create a Scanner that is global (as you did in program10) so that the main() and the reroll() methods can both access it. The methods below are required. Each method should be fewer than 30 lines of code. You may write other methods as needed. Those should also each be fewer than 30 lines of code. All of these should be **public static**

|  |  |
| --- | --- |
| Method | Description |
| **void main(String[] args)** | Creates an int array representing the five dice. Handles the overall flow of the program (asking the user if she wants to play again, for example). Doe not ask the user which dice to reroll (that happens in the reroll method). |
| **void roll(int[] dice)** | Rolls all five dice by assigning a random integer (1 - 6 inclusive) to each of the elements in the array |
| **void sort(int[] dice)** | Implements the bubble sort algorithm to sort the dice in ascending order (small to large) |
| **String toString(int[] dice)** | Returns a space-separated list of the dice in the array. Does not do any printing. |
| **void reroll(int[] dice)** | Asks the user to indicate which dice to reroll by entering a 5-character string consisting only of the letters y and n (y meaning yes and n meaning no). The user may only use y and n, and must enter at least one y and exactly 5 characters. Repeatedly reprompt the user until she enters a valid response. Then, reroll only the specified dice. |
| **int yahtzee(int[] dice)** | Returns the score for yahtzee (50 if all 5 dice are the same, and zero otherwise). |
| **int fourOfAKind(int[] dice)** | Returns the score for four of a kind (the total of all five dice, if at least 4 of the dice have the same value, and zero otherwise). Note that if the dice contain a yahtzee, they also contain 4 of a kind. |
| **int threeOfAKind(int[] dice)** | Returns the score for three of a kind (the total of all five dice, if at least 3 of the dice have the same value, and zero otherwise). Note that if the dice contain a yahtzee or 4 of a kind, they also contain 3 of a kind. |
| **int largeStraight(int[] dice)** | Returns 40 if the dice contain a large straight, and zero otherwise. |
| **int smallStraight(int[] dice)** | Returns 30 if the dice contain a small straight, and zero otherwise. Note that if the dice contain a large straight, they also contain a small straight. |
| **int fullHouse(int[] dice)** | Returns 25 if the player has a full house (three dice contain three of a kind, and two of a kind), and zero otherwise. Note that a yahtzee also contains a full house. So, {3, 3, 3, 6, 6} would be considered a full house, as would {3, 3, 3, 3, 3}. |
| **int sum(int[] dice, int key)** | Returns the sum of all the dice whose value match key. For example, if dice is the array containing {2, 3, 4, 4, 5}, then sum(dice, 4) would return 8, sum(dice, 3) would return 3, and sum(dice, 6) would return 0. |
| **int chance(int[] dice)** | Returns the sum of all the dice. |
| **void scoreChoices(int[] dice)** | Prints all the non-zero scoring options for the given dice. The choices should be listed in the order that they appear on a typical Yahtzee score sheet (see below), and should be formatted as shown in the sample run. |
| **int maxScore(int[] dice)** | Returns the highest possible score that can be obtained with the given dice. |

**It is very important to write all methods, with the exact names, return types, parameters, and so on. Even if you do not know how to implement a method, write its stub because your code will be tested with a separate tester class. Failure to write methods according to requirements will result in at least a 50% deduction.**

The score sheet below shows the order of the various roll types in a game of Yahtzee. This order is needed when displaying to the user her various scoring options.



<https://s-media-cache-ak0.pinimg.com/originals/56/49/65/56496502da61a9b81d614dd6f6c6c998.png>

**It is very important to write all methods, with the exact names, return types, parameters, and so on. Even if you do not know how to implement a method, write its stub because your code will be tested with a separate tester class. Failure to write methods according to requirements will result in at least a 50% deduction.**

***The game play:***

Each game will start with the display of the faces of the 5 dice, and all the non-zero scoring options for those dice. Then, the user is given the option for up to two rerolls. Once the user is done rolling, the maximum score for the dice is displayed.

Whenever the dice are displayed, they should be sorted first. Note that sorting the dice also makes many of the other methods easier to implement.

The array will be declared within the main method and must be passed to all supporting methods.

**Submission requirement (2 point deduction for incorrect submission):**

* Create a folder named program12.
* Inside that folder place exactly one file: **YahtzeeGame.java**
* Zip the program12 folder (not just the file in the folder, but the entire folder).
* Name the zip file program12.zip.

**Sample run #1: Notice that as soon as the user quits, her score is given (this will be the maximum of all the possible scores) on the current dice. Notice what happens if the user does not respond with y or n:**

|  |
| --- |
| Welcome to Yahtzee!  Roll #1: 3 3 3 3 6  3 total: 12  6 total: 6  3 of a kind: 18  4 of a kind: 18  chance: 18  Roll again? **yes**  Roll again? **no**  Roll again? **n**  Your score is 18. Goodbye. |

**Sample run #2: Notice that the user can have up to three rolls total (the initial roll and up to two rerolls). After the three rolls, the game automatically ends, reporting the score. Notice what happens when the user enters an incorrect list of letters for the reroll.**

|  |
| --- |
| Welcome to Yahtzee!  Roll #1: 1 1 2 3 4  1 total: 2  2 total: 2  3 total: 3  4 total: 4  small straight: 30  chance: 11  Roll again? **y**  Indicate which dice to roll using y and n: **nnnnn**  Indicate which dice to roll using y and n: **ynnnnyy**  Indicate which dice to roll using y and n: **ynynQ**  Indicate which dice to roll using y and n: **nnyyy**  Roll #2: 1 1 1 5 5  1 total: 3  5 total: 10  3 of a kind: 13  full house: 25  chance: 13  Roll again? **y**  Indicate which dice to roll using y and n: **nnnyy**  Roll #3: 1 1 1 1 1  1 total: 5  3 of a kind: 5  4 of a kind: 5  full house: 25  chance: 5  yahtzee: 50  Your score is 50. Goodbye. |