**Purpose : (One or 3 member)**

The purpose of this assignment is to help you review some of the main topics covered in previous courses, such as classes, loops, arrays, static attributes and static methods.

**General Guidelines When Writing Programs:**

* Include the following comments at the top of your source codes

// -----------------------------------------------------

// Assignment (include number)

// Question: (include question/part number, if applicable)

// Written by: (include your name and student id)

// -----------------------------------------------------

* In a comment, give a general explanation of what your program does. As the programming questions get more complex, the explanations will get lengthier.
* Include comments in your program describing the main steps in your program.
* Display a welcome message which includes your name(s).
* Display clear prompts for users when you are expecting the user to enter data from the keyboard.
* All output should be displayed with clear messages and in an easy to read format.
* End your program with a closing message so that the user knows that the program has terminated.

**Part I)**

In this part, you need to create a class called LadderAndSnake. A LadderAndSnake object has a board of 10x10 and a number of players attached to it, which is initialized at the creation time of the object. The number of players must be between 2 and 4 inclusively. The number of players must be set through the use of a parameterized constructor of the class. In addition, the board itself is set as shown in Figure 1 below.

A picture containing text

Description automatically generated

**1 = 38**

**4 = 14**

**9 = 31**

**16 = 6**

**21=42**

**28 = 84**

**36=44**

**48=30**

**51=67**

**62=19**

**64=60**

**71=91**

**80=100**

**93=68**

**95=24**

**97=76**

**98=78**

Besides the constructors, and all basic methods in the class, the class must include two methods, one called flipDice(), which should return a random value between 1 and 6 inclusively. The other method is called play(), which actually initiate the core engine of the game where the players start to play the game. The rules of the game are as below:

* Before any of the players starts playing, the order of playing turns must be determined. For that, each player must throw the dice to obtain the largest possible number. In case of a tie between any of the players, the process is repeated only between those players. This process is concluded once the order of playing is determined.
* At this point, the players start playing the game by alternating dice flipping.
* Each dice flip will move a player form square 0 (which you can think about it as outside the board) with the value of the dice. For example, if a player is at square 0 and the dice value was 5, then the player moves to square 5.
* If the reached square has a bottom of a ladder, then the player moves up to the square that has the top of the ladder. For instance, if a player is at square 33, and the flipped dice value was 3, then the player moves to square 36, which in turn will end moving the player up to square 44.
* If the reached square has a head of a snake, then the player moves down the square that has the tail of the snake. For instance, if a player is at square 77, and the flipped dice value 2, then the player moves to square 79 (which has the tip of the snake’s head), which in turn end moving the player down to square 19.

**You will have to find a way to record the relation between these particular ladder and snake squares**

* The game is concluded once any of the players EXACTLY, reaches square 100.
* If a player is close to 100, and the dice value exceeds the maximum possible moves, the player moves backward with the excessive amount. For instance, if a player is at square 96 and the dice value is 5, then the player moves to 99 (that is 4 moves to 100, then 1 move backward to 99).

As a general requirement, you must show/display ALL operations of the game. For instance, at start, you should indicate something like:

* Game is Played by *x* players (where *x* is the number of players set for that game);
* Now deciding which player will start playing;
* Player 1 got a dice value of 5

Player 2 got a dice value of 3

Player 3 got a dice value of 3

A tie was achieved between Player 2 and Player 3. Attempting to break the tie

Player 2 got a dice value of 2

Player 3 got a dice value of 6

* Reached final decision on order of playing: Player 1, Player 3, Player 2
* Player 1 got a dice value of 5; now in square 5
* Player 3 got a dice value of 6; now in square 6
* Player 2 got a dice value of 4; gone to square 4 then up to square 14
* Game not over; flipping again
* Player 1 got dice value of 4; gone to square 9 then up to square 31
* Player 3 got dice value of 4; now in square 10
* Player 2 got dice value of 2; gone to square 16 then down to square 6
* Game not over; flipping again
* :
* :

The above sample of displays is the basic minimum that you should have. A better mark is given for implementing the board as a 2-D array, and for a more elaborate/creative displays (i.e. possibly visibly showing the plays/moves, giving each of the players a name, etc. There is no limit of what you can do here. Be impressive!).

**Part II)**

Create a public driver class called PlayLadderAndSnake. In the main() method, you need to prompt the user to enter the number of players that he/she needs to play the game with. The user must enter a value between 2 and 4. If the user does not enter a correct value, the program must prompt the user again to enter a correct value. However, there is a limit of 4 total attempts. If the user enters an incorrect value 4 times, then the program must display a message to this effect and terminates. For instance, the program behaviour may look something similar to the following:

**Text

Description automatically generated**

* Of course entering a good value before the number of attempts is exhausted will result in the game being played until one player wins.
* final note: You can see that the description of the assignment, while surely sufficient, is very undetailed! Be creative; this is programming!