EECE 230 INTRODUCTION TO PROGRAMMING
Lab Assignment 8

1. *Snakes-and-ladders* is an ancient board game in which two or more players race to the *finish* tile using dice rolls. Every time a player reaches a *snake* tile, he/she is moved backward a number of tiles. On the other hand, reaching a *ladder* tile causes the player to advance several locations forward.

In this exercise, you need to mimic a *snakes-and-ladders* game with one player and count the number of steps required to reach the *finish* tile. You can assume that there are 100 tiles and that each dice roll results in a random number between 1 and 6. Also, you can assume that snakes are found at tiles that are multiples of 13 and ladders are found at tiles with prime numbers (except 13). The number of steps to move forward (in case of ladders) or backwards (in case of snakes) is a random number between 1 and 10. To do this, you need to implement a class `Game` with the following members:

- Private member `currentLocation` of type `int`
- Default constructor that sets `currentLocation` to 0
- Member function `bool isAtFinishTile()` that returns `true` if the current location is 100 and `false` otherwise
- Member function `bool isAtLadderTile()` that returns `true` if the current location is a *ladder* tile and `false` otherwise
- Member function `bool isAtSnakesTile()` that returns `true` if the current location is a *snake* tile and `false` otherwise
- Member function `void rollDice()` that generates a random number between 1 and 6 and updates `currentLocation` accordingly
- Member functions `void moveForward()` and `void moveBackward()` that generate a random number between 1 and 10 and update `currentLocation` accordingly. Note that the value of `currentLocation` should never become less than 0 or greater than 100.

Also, you need to write a `main` that instantiates an object of type `Game` and keeps rolling the dice until the player reaches the *finish* tile. Of course, every time the player reaches a *ladder* tile or a *snake* tile, the appropriate action must be done. The program must print the intermediate locations as well as the total number of steps taken to reach the *finish* tile.

2. In this exercise you are going to develop a simple system to manage a DVD rental store.

a. Implement a class `DVDType` with the following members:
   - Private members: `title` (of type `string`), `genre` (of type `string`), and `nbCopiesAvailable` (of type `int`)
   - Default constructor that sets `title` and `genre` to `""` and `nbCopiesAvailable` to `0`
   - Member functions: `string getTitle()`, `string getGenre()`, and `int getNbCopiesAvailable()`
   - Member function `void set(string, string, int)` that sets `title`, `genre`, and `nbCopiesAvailable` to the values specified as parameters
   - Member function `void rentDVD()` which decrements the number of copies available
   - Member function `void returnDVD()` which increments the number of copies available

b. Write a program to manage the information related to the DVD store as follows:
   - Declare a named constant `maxSize` as a global variable to represent the max number of DVDs in the store and initialize it to 1000. Note that the number of DVDs might be much less than that.
   - Write a function `void loadData(DVDType A[])` that reads the information about the DVDs from a file called *"dvds.txt"* and stores it in an array `A` whose size is equal to `maxSize`. If the number of DVDs is less than `maxSize`, then the remaining (rightmost) cells in `A` will be considered empty. Each line in the file contains information about one dvd using the following format

     title genre nbCopiesAvailable

     You can assume that the title and the genre contain no spaces.

   - Write a function `void saveData(DVDType A[])` that stores the data of `A` in *"dvds.txt"* using the format described above. Note that you need to discard the empty cells.
   - Write a function `int searchByTitle(string str, DVDType A[])` that returns the index of the DVD in `A` whose title is equal to `str`. It returns -1 if the DVD doesn't exist

- Write a function `void displayByGenre(string str, DVDType A[])` that displays all DVDs in `A` whose genre is equal to `str`.
- At the beginning of `main`, you need to declare an array of `DVDType` and read the information from *"dvds.txt"* using `loadData`. Then, you should repeatedly ask the manager to select one of the following functionalities:
    1. **Search by title**: read a title from the manager and tell whether the corresponding DVD exists in the store
    2. **Display by genre**: read a genre from the manager and display the titles of all DVDs of that genre
    3. **Rent DVD**: read a title from the manager and decrement the number of copies of the corresponding DVD. If the title doesn't exist or if there are no available copies, an error message should be generated.
    4. **Return DVD**: read a title from the manager and increment the number of copies of the corresponding DVD. If the title doesn't exist or if there are no available copies, an error message should be generated.
    5. **Exit**: exit the loop.
- After the main exits the loops and right before termination, the program should save the updated data in *"dvds.txt"* using `saveData`.

3. You are required to redo exercise 1 of assignment 4 using object oriented methodology. As such, you need to do the following:

a. Implement class `Point` that has the following members:
   - Private variables `x` and `y` of type `double` that represent the coordinates of the point.
   - Default constructor that initializes `x` and `y` to random values between 0 and 1.
   - Setters and getters

b. Implement class `Circle` with the following members:
   - Private variables `x` and `y` of type `double` that represent the coordinates of the center.
   - Private variable `R` of type `double` that represents the length of the radius.
   - Default constructor that initializes `x` and `y` to 0 and `R` to 1.
   - Function `bool containsPoint(Point p)` that return `true` if `p` lies inside the circle and `false` otherwise.

c. Write a main that reads a positive integer N from the user and prints an estimate of π using N random points. That is, you have to create one instance of class Circle with center (0,0) and radius 1 and N instances of class Point. For each point you create, you check whether it lies inside the unit circle and update the counter accordingly. If M points lie inside the circle, then the estimate of π would be $4 \times \frac{M}{N}$.