CS146 Data Structures and Algorithms
Fall 2020, Instructor: K. Potika SJSU CS Department

Programming Project 1 (A & B)

Important: Do individually (each student alone is NOT a group assignment), each student has to turn in a java program. For this project we will have Code review with the grader or the Instructor. Copying from other students or other resources is plagiarism.

A. Data Shuffling

Description

You are assigned to implement a data-shuffling program. Given is a dataset in the form of a .txt file, where each line (row) corresponds to a pair of science collaborators (note that the 1st line starting with % has metadata not real data). The output contains the same lines (rows) of collaborators as the input but now the lines are shuffled (aka a permutation of the original dataset).
For the purpose of shuffling use the Fisher–Yates shuffle algorithm that works in O(n) running time and O(1) space, as given next. You have to create pseudo-random numbers (see end for help) in O(1) running time.

Algorithm Fisher-Yates

Given n elements, the basic idea is to start from the last element, swap it with a randomly selected element from the whole dataset. In the next step you will consider the rows from 0 to n-2 (size reduced by 1), and repeat the process until you reach the first element.
Follow the next pseudocode to shuffle:

```
-- To shuffle an array a of n elements (indexes 0..n-1)
 for i=n-1 downto 1 //index starts from 0
       j = random integer within 1 <= j <= i
       exchange a[j] and a[i]
```

Applications

Data masking or obfuscation for security purposes. Machine learning to avoid patterns and bias before the training phase. Deck of cards etc. Can you think of other applications?

Dataset

Use the provided Erdos Collaboration Networks (see more about the mathematician Paul Erdős and the Erdős number). The file is the ErdosCA.txt, the first line contains metadata, i.e., the distinct numbers in each column (2 values) and the total number of rows. The next lines contain the actual data of pairs of collaborators (one pair in each line). Shuffle the lines (order of pairs).

Output

Write a program that uses the provided ErdosCA.txt as input and outputs the shuffled array in a file called *LastNameFirstNameShuffled.txt.*

Count the time to read from file, to shuffle the lines and to create the output file.
Note: To count the time use system.currentTimeMillis().

Testing

Create appropriate JUnits to test your program. Help with JUnits:
Instructions for developing JUnit:

- To compare two text files in Junit, you can try the following code.

Use BufferedReader to read the input files.

```
BufferedReader Out=new BufferedReader (new FileReader (<Path of output file>));
BufferedReader In=new BufferedReader (new FileReader (<Path of input
file>));
while ((expectedLine = In.readLine ()) != null) {
               String actualLine = Out.readLine ();
               assertEquals (expectedLine, actualLine);
            }
```

Help with pseudo-random number generation:
//Set seed value as 20, so that everybody gets the same sequence of random numbers

```
Random r=new Random();
r.setSeed(20);
```

Compare the output file with attached see next:
if you use nextDouble() use Target1.txt to compare

```
      double d = random.nextDouble();
      int j = (int)(d*arr.length);
```

else if you use nextInt() use Target2.txt

B. Circular linked list game

Description

In an ancient land, a King had many prisoners. He decided on the following procedure to determine which prisoner to grand freedom. First, all of the prisoners would be lined up one after the other and assigned numbers. The first prisoner would be number 1, the second number 2, and so on up to the last prisoner, number *n*. Starting at the prisoner in the first position, he would then count k prisoners down the line, and the kth prisoner would be eliminated from winning her/his freedom removed from the line. The King would then continue, counting k more prisoners, and eliminate every kth prisoner. When he reached the end of the line, he would continue counting from the beginning.
For example, if there were six prisoners, the elimination process would proceed as follows (with step k=2):

1->2->3->4->5->6 Initial list of prisoners; start counting from 1.
1->2->4->5->6 Prisoner 3 eliminated; continue counting from 4.
1->2->4->5 Prisoner 6 eliminated; continue counting from 1.
1->2->5 Prisoner 4 eliminated; continue counting from 5.
1->5 Prisoner 2 eliminated; continue counting from 5.
1 Prisoner 5 eliminated; 1 is the lucky winner.

Pseudocode and Output

Write an efficient program that creates a circular singly linked list of nodes to determine which position you should stand in to win your freedom if there are *n* prisoners. Your program should

simulate the elimination process by deleting the node that corresponds to the prisoner that is eliminated for each step in the process.

Count the time to create list, to delete one single node and time to find the winner.
Note: To count the time use system.currentTimeMillis().

Testing

Create appropriate JUnits to test your program.

Help with JUnits:
Instructions for developing JUnit:

1. Upon creation of linked list,
    1. Check if linked list is empty using assertTrue.
    2. Check if length is 0 using assertEquals.
    3. Similarly, after adding elements, linked list should not be empty and size not equal to 0.

Sample code:

```
@Test
public void test() {
      linkedList prisoners=new linkedList();
      assertTrue(prisoners.isEmpty()); //before inserting, list is empty
      assertEquals(0, prisoners.size); // Size is 0
      prisoners.insert(5);
      assertFalse(prisoners.isEmpty()); // after inserting element, list is not
empty
     assertEquals(1,prisoners.size); //size is 1
```

Test cases (n is the number of prisoners, k is the step):

1. For n=6, k = 2, Output: 1
2. For n= 1, k = 9, Output =1
3. For n=7, k = 7, Output = 4
4. For n =12, k = 8, Output= 2
5. For n = 5, k = 1, Output = 3

For both programs:

Programming Standards:

- See project rubric on canvas for details.

Deliverables:

- ✓ For JUnit tests check canvas.
- ✓ Use cs146F20.<lastname>.project1 as your package, and don't need a main method, only JUnit java tests.
- ✓ What to submit to canvas (Steps):
    - Export your project on eclipse with your entire project (source code). Otherwise include a Readme file.
    - Create a zip file named **LastnameFirstProjectName.zip** that includes the exported zip file + pdf report

- upload the last zip file to canvas.

✓ All projects need to compile. If your program does not compile you will receive 0 points on this project.
✓ Add meaningful comments.
✓ Do not use any fancy libraries. We should be able to compile it under standard installs. Include a readme file on how to you compile the project.
✓ Check Programming Project Rubric.