Efi Arazi School of Computer Science

Introduction to Computer Science

2023 Moed Bet: Final Examination

- The exam lasts 3 hours. There will be no time extension.
- Use your time efficiently. If you get stuck somewhere, leave the question and move on to another question.
- Use of digital devices, books, lecture notes, slides, computers, calculators and internet and anything other than this exam form is forbidden. All the material that you need for answering this exam are supplied with the exam.
- Answer all questions on the current exam form using a pen. Write your answers on the front side of the page, the back side of the page won't be scanned and won't be checked. You may use draft pages which will not be checked.
- You can answer any question in either English or Hebrew.
- If you feel a need to make an assumption, you may do so as long as the assumption is reasonable and clearly stated.
- If you can't give a complete answer, you may give a partial answer. A partial answer will award partial points.
- If you are asked to write code and you feel that you can't write it, you may describe what you wish to do, in English or Hebrew. A good explanation will award partial credit.
- If you are asked to write code that operates on some input, there is no need to validate the input unless you are clearly asked to do so. Likewise, if you are asked to write a function that operates on some parameters, there is no need to validate the parameters unless you are explicitly asked to do so.
- There is no need to document the code that you write unless you want to communicate something to us.
- In some of the questions there are constraints, if you chose to ignore the constraint it will allow you to solve the question with less difficulty. If you chose to have the constraint, the grading will start from maximum mark on that question, otherwise the new max grade is mentioned in the constraint.
- The code that you will write in the exam will be judged, among other things, on its conciseness (תמציתיות), elegance, and efficiency. Unnecessarily long or cumbersome code will cause loss of points, even if it provides the correct answer.
- **Don't write anything on the back of the pages**. Only the front pages are scanned for grading. You will get some blank pages for draft (טיוטה).
- **<u>Your hand-writing must be readable. Unreadable answers will get a 0 grade.</u>**

<u>Good luck!!</u>

The first help pages contains several functions from the String class. You may use these functions as needed.

Part A: 36 points (Q1-Q3)

1. 12 points

Given a 2-dimentional array NxN named “`mat`” containing integers (positive, negative and zero), that fulfills the following conditions:

a. Every row is sorted in an increasing way (and there are no two identical numbers in each row)
b. Every column is sorted in an increasing way (and there are no two identical numbers in each column)

Implement the function `numOfNegativeNumbers` whose input is a 2-dimnetional array mat as described above, that returns the number of negative numbers in the array.
For example, for the following matrix the function will return 10.

-90	-86	-55	-9
-89	-85	-16	0
-70	-60	-10	5
10	12	14	15

The required time complexity is linear O(N), where N is the dimension of the matrix. A solution with higher complexity will earn fewer points.

```
public static int numOfNegativeNumbers (int [][] mat)
```

Extra page

2. 10 points

write a **recursive** function whose input is a String str and an integer x, and that returns true if the first x characters in str are symmetric to the last x characters in str; false otherwise.

For example:

isSymmetric(“abdgutyrgdba”,4) // true
isSymmetric(“abdgutyrgdba”,2) // true
isSymmetric(“abca”,2) // false
isSymmetric(“aaa”,2) // true
isSymmetric(“aaaa”,5) // false

```
public static Boolean isSymmetricX(String str, int x)
```

3. 12 points

An integer bigger than 1 will be called k-almost prime if it is composed of exactly k prime factors.
For example:
The number 14 is 2-almost prime since it’s composed of 2 prime factors: 2*7
The number 13 is 1-almost prime since it’s composed of 1 prime factor: 13
The number 100 is 4-almost prime since it’s composed of 4 prime numbers: 5*5*2*2

Implement the kAlmostPrime function (int n, int k) where n>1 k>0. The function returns true if n is a k-almost prime, false otherwise.

```
public static boolean kAlmostPrime(int n, int k)
```

Extra page

Part B: 64 points (Q4-Q12)

Questions 4– 12 are dealing with Object oriented programming.
The questions in this part of the examination focus on a model that consists of 3 Main Objects:

A) **Worker:** has a lower case name of type String, and a salary of type int (which should always be positive), both of which have a protected encapsulation setting.
B) **Manager:** is a type of Worker, therefore Manager **inherits** from Worker. The Manager class has an additional field named workers of type LinkedList<Worker>. The list of workers is sorted lexicographically by name.
C) **Office**: has 3 fields: name (String), location(String) and manager (Manager).

In the following questions, you will be asked to implement several methods in each of those classes.

Q4-> Worker

Q5-Q7-> Manager

Q8-Q12-> Office + Manager

Some of the methods of those classes will be given as API, alongside API for, LinkedList<E> and ListIterator<E> as part of the documentation given to you.

NOTE: NODE IS NOT GIVEN AS API AND EVEN THOUGH IT IS INTEGRAL PART OF LINKEDLIST YOU CANT USE IT (SIDE NOTE : YOU DON'T NEED IT).

EACH NODE USAGE WILL CAUSE A DISQUALIFICATION OF THE ENTIRE QUESTION

<u>Question 4 is dealing with the Worker Class, you should implement the following method:</u>

4. (7 points)
    a. The method `raiseSalary(int amount)`. If the amount is negative, it should cause the runtime error IllegalArgumentException.

```
public void raiseSalary(int amount)
```

    b. What problem can arise if a Worker gets a raise (one or more) whose sum if very big (billions)? And how it can be solved?

<u>Questions 5-7 are dealing with the Manager Class, you should implement the following methods:</u>

5. (5 points)

The constructor `Manager(String name, int salary, Worker[] workers)` builds an instance of `Manager` which builds and insert all of the workers to work directly under the manager.

Implement the constructor.

```
public Manager (String name, int salary, Worker[] workers)
```

6. (12 points)
The `getTotalSalaries()` method returns the total salaries of all the Workers that work under the manager (directly **AND** not directly) **<u>not including the manager</u>**. Implement the method.
<u>Note</u>: Consider the case when Manager has a Manager as one of its workers.
<u>Tip: use recursion and a loop</u>

```
public long getTotalSalaries()
```

7. (5 points)
Implement the `equals` method of the Manager class.

```
public boolean equals(Object obj)
```

Questions 8-12 are dealing with the Office and Manger Classes.

8. (14 points)
The `getWorkersManager(Worker worker)` method returns the given workers direct manager.
Implement the method.
Tip: `use recursion and a loop.`

```
public Manager getWorkersManager(Worker worker)
```

IMPLEMENT ONE OF THE QUESTIONS 9 OR 10. Regardless of your choice, note that you will have to use both of them in the following questions.

9. (9 points) – **choose this question, OR Q10**
The `addWorkerToManager (Worker worker, Manager newManager)` method adds a Worker to that manager. The worker should be added to the list of workers in the correct lexicographic ordered location (by their name). You can assume no two workers have the same name. As part of the implementation, you should implement the addWorker method in the Manager class as well.
Note: you can assume that the worker and the manager are not the same AND the worker is not assigned to any other manager.

```
// Manager
public void addWorker (Worker worker)
```

```
// Office
public void addWorkerToManager (Worker worker)
```

10. (9 points)
The `removeWorkerFromOffice (Worker worker)` method removes the worker from the office. As part of the implementation, you should implement the removeWorker method in the Manager class as well.

Note: you can assume that the worker given is not the manager of the office AND the worker is not a manager with workers.

```
// Manager
public void removeWorker (Worker worker)
```

```
// Manager
public void removeWorkerFromOffice (Worker worker)
```

11. (5 points)
The `getTotalSalaries()` method returns the total of the salaries including the manager of the Office

```
public long getTotalSalaries()
```

12. (5 points)
The `void moveWorkerBetweenManagers(Worker worker, Manager newManager)` method moves a worker from one manager to another.

```
   public void moveWorkerBetweenManagers(Worker worker, Manager
newManager)
```

Extra page

Extra page