Philadelphia Classic 2013 Fall
Hosted by the Dining Philosophers
University of Pennsylvania

dining philosophers

UNIVERSITY OF PENNSYLVANIA | COMPUTER SCIENCE CLUB

This document includes 13 questions. Novice teams do 1-9; standard teams do 6-13. Any team which submits a correct solution for any of problems 1-5 will become a novice team. If you are not a novice team, please do not attempt problems 1-5.

You can only use the internet for PC^2 (to submit your solutions), Javadocs, and referring to any documents we link you to.

Do not modify any of the given methods! They are all specified in the desired format.

There is no penalty for incorrect submissions. You will receive 1 point per problem solved. Number of incorrect submissions will be used only as a tiebreaker.

Some really scary aliens have decided to invade Earth! Oh no! They have super duper advanced technology, but they also have a lot of weird quirks which we can exploit. You need to use your incredible programming skills to help humanity fight them off!

If you are not a novice team, please skip down to problem 6 now.

1. Weird Repeat

One of your brilliant scientists have a theory that the alien language does not contain any two similar adjacent characters. This means that the alien language cannot have any character of the alphabet next to each other (like aaron because of the a's).

Your mission is to help this mad, yet brilliant scientist test his theory by writing a Java function that takes in an input String indexed 1, 2,...., n where n is the size of the input string. It then returns a String in which each character is immediately repeated by the character's index number. Therefore something a word as simple as abc becomes abbccc, which can then be used to confuse the aliens.

Sample Input:
1). abc
2). abc1
3. Eagles!

Sample Output:
1). abbccc
2). abbccc1111
3). Eaaggglllleeeeessssss!!!!!!!

2. Soldiers' Wages

The Earth's Soldiers have been working hard to fight off the aliens and now it's time to pay them. Bureaucratic nonsense in Washington (The command center for the Earth's war effort for obvious reasons) has led to the rules for paying soldiers being obnoxiously convoluted. Here they are:

1.) The standard hourly wage: $10.00/h
2.) Any hours over 8 a day get an overtime rate: $12.00/h
3.) If the soldier works on Sunday they get a higher rate as well (Note: The overtime rate does not apply on Sunday): $15.00
4.) If the soldier works on Saturday they get the same deal as with Sunday but a different rate: $13.00
5.) If at the end of the week a soldier has worked more than 40 hours he/she gets an extra dollar for each of those hours.

Sample Input:

You will be given an array of 7 integers representing the number of hours worked each day in the order Sunday-Saturday.

1.) 0 5 9 2 2 3 2
2.) 2 10 8 2 9 10 9

Sample Output:

Return the integer value of the number dollars a soldier earns that weak.

1.) 238
2.) 557

3. Is Palindrome Prime

The aliens trying to invade Earth are really superstitious and consider it really bad luck to invade coordinates that are both palindromes and primes. So, naturally, Earth's commanders think it a good idea to set up headquarters in one such coordinate.

Your mission, should you choose to accept it, is to write a Java method that takes an int as an input and return a boolean value indicating whether the int is a palindromic prime or not. FYI, the largest palindromic prime known is 10314727 - (8 x 10 ^ 157363) - 1, and if the memory and runtime limitation of your program does not exist (in an ideal world), your code should work on this.

Sample Input:

1.) 131
2.) 1
3.) 121
4.) 1234

Sample Output:

1.) true
2.) true
3.) false
4.) false

4. Alien DNA

Our best researches have been analysing the DNA of the aliens and have found something interesting when trying to synthetically generate some alien DNA. Alien’s DNA has two regular bases A, and T just like ours, but the other two bases operate in a way similar to the way brackets work in our language. If the scientists try and make a DNA string with either open brackets or extra closing brackets the DNA is invalid and can’t grow a new Alien. You’ve been asked to write a program that determines if a given DNA sequence is valid.

Sample Input:
You will be given a string with the DNA string as follows

1.) AATT[A]A[A]]
2.) AATT
3.) A[T][A][[A]]

Sample Output:
You should return INVALID if the DNA contains some error and VALID if it is written fine

1.) INVALID
2.) VALID
3.) VALID

5. Food Drops

Some parts of the war effort aren’t glamorous. You are a soldier charged with keeping track of the food drops for your platoon. Unfortunately the list of flights and arrival times isn’t sorted so it’s a pain to figure out what flight is next. Fortunately you are proficient in java and decided to write a program to put your list in order from earliest to latest.

Sample Input:
The input will be in the form of two arrays of equal length. The first array is an array of strings that represent the names of the flights. The second is an integer array that represents the times the flights arrive in army time (you don’t have to worry about AM PM).

1.) [“USA123”, “UPS549”, “UAF109”] [1635, 1210, 0650]
2.) [“DAL456”, “AFR333”, “RCH007”] [0230, 0670, 2355]

Sample Output:
You will output the array of sorted flight names encoded as a string in the following format: brackets enclosing the list, and a comma and space after each element except for the last one.
1.) [UAF109, UPS549, USA123]
2.) [DAL456, AFR333, RCH007]

6. Force Field Containment

Oh no! You and your secret spy group sent into known alien controlled territory have been discovered. The aliens have somehow suspect that they have spies somewhere in their territory or near their territory. You know your coordinates, and somehow, command headquarters was able to send you the coordinates of the force field the aliens put up to trap you guys. You now need to figure out if you and your group are trapped in this force field or not.

Your survival now depends on you being able to tell if you are contained within this force field. Therefore, write a Java method that takes in 2 inputs and tells whether or not you are trapped in the force field. You will be provided with an inner class called Point which holds the x and y position of a point. The first input will be an object type Point that will contain your location. The second input will be a Point array that will contain the locations of the beams that were used to set up the force field. These locations, given in clockwise order, form a polygon that makes up the force field. Your method should return a boolean value that indicates whether or not you're trapped.

Sample Input:
1.) Point (0, 0), { Point (-1, -1), Point (-1, 1), Point (1, 1), Point (1, -1) }
2.) Point (5, 0), { Point (-1, -1), Point (-1, 1), Point (1, 1), Point (1, -1) }
3.) Point (0.5, 0.5), { Point (0, 0), Point (1, 1), Point (1, 0) }
4.) Point (0, 0), { Point (0, 0), Point (1, 1), Point (1, 0) }
5.) Point (-1, 0), { Point (0, 0), Point (1, 1), Point (1, 0) }
Sample Output:
1.) True
2.) False
3.) True
4.) True
5.) False

7. Tallest Tower

The aliens are coming to assault your city! You need to build a tower to defend your city. You would like the tower to be as tall as possible, but the aliens will build a ladder to try to scale your tower. To do so, the aliens will connect smaller ladders at each end. The aliens have an infinite number of smaller ladders, but they only come in two sizes, X & Y (Note: X is always larger than Y). If the aliens ladder is too tall (i.e. taller than your tower), you can push it off the tower. If it is too short, it obviously will not be able to reach your tower. Your goal is: given the sizes of their smaller towers, output the largest tower that you can build that the aliens will not be able to scale. You can output -1 if you could build an infinite height tower.

Sample Input:
1.) 2, 3
2.) 3, 6
3.)1, 2
4.) 4, 3

Sample Output:
1.) 1
2.) -1
3.) 0
4.) 5

8. Planetary Projectiles

The aliens are inside the solar system! They've started firing their lasers at Earth but fortunately they're not entirely sure where it is so they're just firing randomly. Also, we've cracked their code so we know when and and in what direction they're going to fire. Given the positions and sizes of all planetary objects including Earth (represented as circles with position and radius) and the position and firing direction it's your job to figure out if the laser is going to hit earth, miss everything, or hit another planet before hitting earth. (Fortunately the ship is in the orbital plane so we can pretend this all happens in a 2 dimensional surface).

First you will be given the number of planets then the coordinates and radius of each starting with Earth. The information for the planet will be given in the form x y r. Finally you will be given the coordinates of the alien ship and the direction in vector form that it will fire in the form x y vx vy where vx and vy are the components of the laser's "velocity vector" (it points in the direction the laser fires) (Note, we assume that the Earth is not moving). You should output one of the following strings: "HIT EARTH", "MISS", "HIT PLANET".

Sample Input:
1.) 2 10 10 0.5 13 13 0.6 15 15 -1 -1
2.) 2 9 19 0.5 10 12 0.6 9 16 0 -1
3.) 2 10 12 0.5 16 12 0.6 4 11 10 1
Sample Output:
1.) HIT PLANET
2.) MISS
3.) HIT EARTH

9. Aliens Among Us

You've learned that, before the invasion, the aliens planted secret agents on Earth to spy on and sabotage human defenses. While these agents seem indistinguishable from humans, you've learned that they have a poor understanding of Earthly culture. Thus, you decide to try to see if people are aliens by asking a series of culturally related questions. After much sampling, you discover that true humans who live near each other usually give similar answers to cultural questions. Therefore, you decide to figure out who the aliens are by using the following method.

- Sample many people in a region
- Compute the modal (i.e. most common) answer to each question
- Count the number of deviations (i.e. differences) that each person has from the modal answers
- Conclude the person with the maximum number of deviations is the alien

Input Format: A two by two rectangular String array where each column is a name followed by that person's answer. (eg. for the sample data[1][0] = "Jane").

Output Format: The name of the alien

Guarantees:
- There will never be a "tie" for whom the alien is
- There will always be a modal response for each answer
- No two "people" have the same name
- Everyone will be asked the same number of Q's
- There will always be at least one person who will be asked at least one question

Sample input:

1.) Bob, Philly, Eagles, Invincible, Penn, Cheese Steak
Jane, Philly, 76's, Invincible, Penn, Pretzel
Joe, Camden, Giants, Million Dollar Baby, Princeton, Cheese Steak
Serena, Philly, Eagles, Ali, Penn, Pasta

Sample Output:

Notice: You don't need to know what the questions were.
In this case, the modal responses are "Philly", "Eagles", "Invincible", "Penn", "Cheese Steak". So the number of deviations are (in order): 0, 2, 4, 2, so Joe has the most deviations, so we conclude he is the alien and output:
1.) Joe

If you’re a novice team, stop here! If you’re not, you’re in luck - they get harder now!

10. Alien Royalty

As a study of alien history, you learned that there is one lead monarch of the alien civilization. Alien monarchs, much like historical European monarchs, enjoy naming their children after them. (Think of every Henry of England or Louis of France!)

An alien name constitutes a one-word name, a space, and a Roman numeral-esque number. Alien numerals function identically to Roman numerals with a couple of changes. The letter X corresponds to 10 in both Roman and alien numerals. However, the letter Y corresponds to 5 in alien numerals (as opposed to V), and the letter Z corresponds to 1 in alien numerals (as opposed to I). You can assume that all alien numerals are valid and that you will see no numeral characters besides X, Y, and Z.

As a review of Roman numeral rules, characters are placed in descending order of value. One cannot use four or more consecutive identical characters. For example, IV represents 4, not IIII. Likewise, IX represents 9, not VIIII.

You will be given a comma-delimited String of alien monarch names. Your job is to sort them and return a comma-delimited String of alien monarch names. Alien names are sorted alphabetically, with ties being broken by chronological order (i.e. the fourth king should come before the fifth king of the same name).

Sample Input:
1.) “Gilgamesh Z,Cthulu ZZ”
2.) “Gilgamesh XY,Gilgamesh X,Cthulu Z”

Sample Output:
1.) “Cthulu ZZ,Gilgamesh Z”
2.) “Cthulu Z,Gilgamesh X,Gilgamesh XY”

11.) Find the General

You have discovered some very important war changing information so you must get to your commanding officer ASAP to give him the info. You are currently on an asteroid, and you know the asteroid your commanding officer is on. You are able to jump from some asteroids to other ones through some portals. However, you also know that there are other asteroids controlled by the aliens which you want to avoid.

Therefore you need to write a Java function that will determine the shortest, and therefore the fastest, path from you to your commanding officer avoiding alien asteroids at all times.
You will be provided an inner class Asteroid that describes the asteroid. Each asteroid object has a field indicated its ID (each asteroid has a unique ID), a boolean field indicating whether or not it is an alien occupied asteroid and also a List of Integers indicating what asteroids it's connected to.

Your method will be given 3 inputs: Asteroid source, Asteroid target and a List of Asteroids in the system. Your program must take those inputs and return the shortest path from you to commanding officer without going to any alien controlled asteroids.

Sample Input:

```
1,5
1,false,2,3
2,true,1,3,5
3,false,1,4
4,false,3,5
5,false,2,4
```

In this example you are on asteroid 1, trying to move to asteroid 5. The next 5 lines give you the ID of the asteroid (always an integer) followed by a boolean indicating if it's alien controlled. The next few integers in each line say which asteroids that asteroid is connected to. You won't have to parse this info, it will be given to you in the arguments of the method we provide you.

Sample Output:

```
3
```

12.) Spotting Ships

In previous combat with the aliens, we noticed that they had a strange inability to target ships painted in a specific pattern of camouflage - if the entire ship is green with exactly 2 brown spots, the aliens somehow were unable to hit us!

As a result, we painted all our ships in this manner. However, it seems the aliens have adapted. New intelligence suggests that while they can now target 2-spot ships, they will not be able to target ships painted with exactly 1 spot.

We want to defend our ships by painting them in a way that merges their two spots into one. The current paint on the ship is represented by an N by M (1 <= N, M <=50) grid of characters like this:

```
................
..XXXX....XXX...
...XXXX....XX...
.XXXX......XXX..
........XXXXX...
.........XXX....
```

where each '.' is green paint, and each 'X' is part of a brown spot. Two 'X's are part of the same spot if they are vertically or horizontally adjacent (diagonally adjacent does not count).

Being on a tight budget, we want to use as little paint as possible to merge the two spots into one. In the example above, we can do so by painting only 3 additional sections with brown paint (the new brown spots are marked with '*'s below):

```
................
..XXXX....XXX...
...XXXX*...XX...
.XXXX..**..XXX..
........XXXXX...
.........XXX....
```

Help us determine the minimum number of new 'X's to paint in order to merge the two spots.

Input format:
 ints N and M - the dimensions of the grid (N rows, M columns)
 Char array representing which spots are painted (‘X’ means painted ‘.’ means not painted)
  - The grid will always have two distinct spots

Sample Input:

N = 5
M = 16
```
..XXXX....XXX...
...XXXX....XX...
.XXXX......XXX..
........XXXXX...
.........XXX....
```

Sample Output:

3
You only need to output the number but here is one way to fill in the ship with new spots (The ‘*’s are the new spots)
```
..XXXX....XXX...
...XXXX*...XX...
.XXXX..*...XXX..
.......*XXXXX...
.........XXX....
```

13. Mercy!

We've just learned that the aliens have a bizarre affinity for number theory. As result of this, they've decided to ask us a single question: If we can get this question correct, the aliens will have mercy on us and cease and desist all attacks. The question is as follows:

First, take a list of size n which contains the first n integers (starting at 1) in some permutation. (example: n = 6; [5, 3, 1, 2, 6, 4])

Go through the list for the first k integers, where k ranges from 1 to n. For every level of k, compute the lcm of the first k integers. We do not care what specifically the lcm is, but we will count the number of times the lcm changes.

example for list above:

k = 1; lcm = 5; changes = 1
k = 2; lcm = 15; changes = 2
k = 3; lcm = 15; changes = 2
k = 4; lcm = 30; changes = 3
k = 5; lcm = 30; changes = 3
k = 6; lcm = 60; changes = 4

As you probably noticed, the number of lcm changes is not the same for every permutation of n. In fact, for every n, there is a maximum and minimum number of lcm changes possible, and at least one permutation leading to each. Your mission, should you choose to accept it, FOR THE FATE OF HUMANITY, is to complete the *difference* between the maximum and minimum number of lcm changes in every possible permutation of n elements. But be careful - n can be as large as 10^12!

Sample Input:

1.) 10
2.) 6
3.) 1000

Sample Output:

1.) 4
2.) 2
3.) 26