Robotique
CRC
Robotics

CRC

PROGRAMMING COMPETITION

PRELIMINARY PROBLEM 3

A FEW NOTES

- The complete rules are in section 4 of the rulebook.
- You have until **Sunday, December 15th, 11:59 pm** to submit your code.
- Feel free to use the programming forum on the CRC discord to ask questions and discuss the problem with other teams. It is there for that purpose!
- **We are giving you quick and easy-to-use template files for your code and the tests. You are required to use them.**

USING THE TEMPLATE FILE

- The template tests call the function to test, take the output and allow you to quickly check if your code works as intended. **All your code, except additional functions you create, should be written in the function of the part of the problem you are solving.**
- Points given in the document are indications of how difficult the section is and how many points you will get if you complete it. This preliminary problem is going to be 2% of the main challenge towards the global score of the programming competition and for more points related information consult the rulebook.

STRUCTURE

Every problem contains a small introduction like this about the basics of the problem and what is required to solve it.

Input and output specification:

Contains the inputs and their format, and which outputs the code is required to produce and in what format they shall be.

Sample input and output:

Contains a sample input, sometimes containing sub examples in the sample input, and what your program should return as an output.

Explanation of the first output:

Explains briefly the logic that was used to reach the first output, given the first input.

CRC Hospital

One of the most stressful aspects of CRC is making quick repairs on the robot in the few minutes before the next match. As the entrepreneur you are, you decide to found the CRC Hospital, the very first hospital dedicated to robots!

Running a hospital naturally comes with its share of challenges, but fortunately for you, some of them can be solved with programming. By leveraging programming in your modern hospital, you will become the first facility dedicated to the health and repair of robots.

Part 1: Recessive traits (40 points)

An organism's complete set of genes makes up its genome. A gene is a piece of DNA that codes for a particular characteristic of the organism, for example its eye colour. Each gene is present in two copies, called **alleles**, each of which is transmitted by one parent.

Alleles can be dominant or recessive. A **dominant** allele masks the other allele's ability to express itself, i.e. only one copy of a dominant allele is needed for its trait to be expressed. A **recessive** allele is only expressed if it is present in two copies. We represent a dominant allele by an uppercase letter, and a recessive allele by the same letter but lowercase.

Let's take eye colour as an example. The brown allele (B) is dominant while the blue allele (b) is recessive. So, if someone has a brown allele and a blue allele (Bb), they will have brown eyes.

Now, let's say one parent has the alleles Bb, and the other parent has bb. We can predict the genes resulting from a cross between them with the help of a **Punnett square**:

	b	**b**
B	Bb	Bb
b	bb	bb

Their offspring thus have a 50% chance of inheriting blue eyes!

In this problem, you will analyze (upside down) family trees and recessive diseases in order to calculate the probability *P* of a descendent inheriting a recessive disease.

Here are some formulas for calculating probabilities (read this article to learn more):

$$P_{recessive\ Parent} = P_{inherit\ XX} \cdot P_{transmit\ recessive\ allele} \quad (1)$$

$$P_{recessive} = P_{recessive\ Parent\ 1} \cdot P_{recessive\ Parent\ 2} \quad (2)$$

Input and output specification:

As input you will receive:

- **tree:** a perfect binary tree. Each node of the tree contains two letters representing the node's alleles, and unknown alleles are denoted by "XX". The root (which will always be "XX") represents the person whose genes we are analyzing. The child nodes of the root are the parents of this person, then the child nodes of the parents are the grandparents, and so on.

As output you need to return:

- a *float* variable with a 3 decimal precision, corresponding to the probability in percentage that the root node inherits the recessive disease.

Sample input:

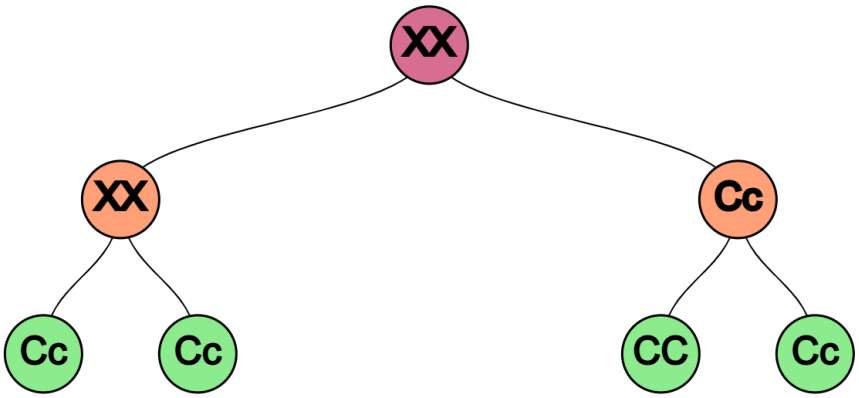
```
["XX", "XX", "Cc", "Cc", "Cc", "CC", "Cc"]

["XX", "XX", "XX", "Ss", "ss", "Ss", "SS"]

["XX", "XX", "XX", "XX", "Tt", "tt", "XX", "TT", "TT", "Tt",
"Tt",
 "Tt", "XX", "TT", "tt"]
```

Sample output:

```
25.0
18.75
18.75
```

Explanation of the first output:

Here is the binary tree we need to analyze:



We can create a Punnett square to find the possible genes that the unknown orange node on the left (which has the alleles “XX”) could have:

	C	c
C	CC	Cc
c	Cc	cc

Method 1: Punnett squares

The unknown parent “XX” could have CC, Cc, Cc, or cc as its alleles. Here are the 4 possible Punnett squares for the alleles of the root node:

	C	C
C	CC	CC
c	Cc	Cc

1

	C	c
C	CC	Cc
c	Cc	cc

2

	C	c
C	CC	Cc
c	Cc	cc

3

	c	c
C	Cc	Cc
c	cc	cc

4

The root inherits the recessive disease if it receives the allele pair **cc**, which corresponds to 4 out of 16 of the boxes, so the probability is $\frac{4}{16} = 25.0\%$.

Method 2: formulas

Since the recessive allele c can't be inherited from CC, it's enough to just consider the two other possibilities, Cc and cc, for parent 1 on the left. We already know the alleles of parent 2 on the right are Cc, so there are two possible combinations: if the parents are Cc and Cc, or if they are cc and Cc.

The probability of inheriting the recessive disease will be the sum of P_1 and P_2, where P_1 is the probability of inheriting the disease from a cross between Cc and Cc, and P_2 is the probability when the cross is between cc and Cc.

Let's examine the cross between Cc and Cc. We want to calculate the probability that each of the parents transmits the c allele. Formula (1) gives us:

$$P_{recessive\ Parent\ 1} = P_{recessive\ Cc\ (left)} = P_{inherit\ Cc} \cdot P_{transmit\ allele\ c} = \frac{2}{4} \cdot \frac{1}{2} = \frac{1}{4}$$

(There is a 2 in 4 chance that parent 1 has the alleles Cc, then a 1 in 2 chance that parent 1 transmits the recessive allele c and not the dominant allele C.)

$$P_{recessive\ Parent\ 2} = P_{recessive\ Cc\ (right)} = P_{inherit\ Cc} \cdot P_{transmit\ allele\ c} = 1 \cdot \frac{1}{2} = \frac{1}{2}$$

(Parent 1 has the alleles Cc, so there is a 1 in 2 chance that it transmits the recessive allele c.)

Then, we use formula (2):

$$P_1 = P_{recessive\ Parent\ 1} \cdot P_{recessive\ Parent\ 2} = \frac{1}{4} \times \frac{1}{2} = \frac{1}{8}$$

Similarly, let's examine the cross between cc and Cc:

$$P_{recessive\ Parent\ 1} = P_{recessive\ cc\ (left)} = P_{inherit\ cc} \cdot P_{transmit\ allele\ c} = \frac{1}{4} \cdot \frac{2}{2} = \frac{1}{4}$$

$$P_{recessive\ Parent\ 2} = P_{recessive\ Cc\ (right)} = P_{inherit\ Cc} \cdot P_{transmit\ allele\ c} = 1 \cdot \frac{1}{2} = \frac{1}{2}$$

$$P_2 = P_{recessive\ Parent\ 1} \cdot P_{recessive\ Parent\ 2} = \frac{1}{4} \times \frac{1}{2} = \frac{1}{8}$$

Finally, according to the sum rule, the probability that the root node inherits the recessive disease is the sum of P_1 and P_2:

$$P = P_1 + P_2 = \frac{1}{8} + \frac{1}{8} = 0.25 = 25.0\%$$

Part 2: Waiting time (20 points)

In your hospital, you have a limited number of repair doctors, and you are trying to determine, based on your number of clients and the average repair time, how long your clients will wait on average. Therefore, you will need to calculate the average waiting time. To help you, here are some explanations and useful formulas from queueing theory.

$\lambda = number\ of\ clients\ arriving\ per\ hour$
$\mu = number\ of\ clients\ that\ can\ be\ treated\ per\ hour\ by\ a\ repair\ doctor$
$c = number\ of\ repair\ doctors$
$\rho = arrival\ to\ treatment\ ratio$

$$\rho = \frac{\lambda}{c \cdot \mu}$$

The arrival-to-treatment ratio is a critical metric. If the ratio value is greater than 1, it means the system is unstable, and arrivals exceed treatments. Consequently, treatment times become infinitely long as time progresses and clients accumulate. In our examples, the ratio will always be less than 1.

$L = number\ of\ clients\ in\ the\ hospital$
$L_q = number\ of\ clients\ waiting\ in\ line$
$P_{full}(\rho, c) = Probability\ that\ all\ the\ repair\ doctors\ are\ occupied$

The probability $P_{full}(\rho, c)$ is a dictionary of values that depend on the ratio ρ\rhoρ and the number of repair doctors ccc. You will be provided with the dictionary, but you will need to find the correct value based on the ρ\rhoρ and ccc of the situation. **The table starts at 2 repair doctors and goes up to 7**. The keys of the dictionary correspond to the ρ values, and all the examples provided will have a ρ\rhoρ that matches an entry in the given table. Once you identify the array associated with the correct ρ\rhoρ, you can find the probabilities of all repair doctors being occupied for 2 to 7 doctors.

```
proba_full = {
    .10: [.02, .00, .00, .00, .00, .00],
    .20: [.07, .02, .00, .00, .00, .00],
    …
}
```

$$L_q = P_{full}(\rho,\ c) \cdot \frac{\rho}{1-\rho}$$

$$L = \frac{\lambda}{\mu} + L_q$$

$W = average\ time\ spent\ in\ the\ hospital\ (waiting\ and\ being\ treated)$

$$W = \frac{L}{\lambda}$$

Input and output specification:

For input you will receive:

- arrivals: an *int* of the number of clients arriving per hour
- mu: an *int* of the number of clients a repair doctor can treat per hour
- c: an *int* of the number of working repair doctors

As an output, you will need to return a *float* with a 2 decimal precision of the average waiting time **in minutes**.

Sample input:

```
arrivals = 80, mu = 50, c = 2
arrivals = 42, mu = 8, c = 7
arrivals = 6, mu = 3, c = 4
```

Sample output:

```
3.33
9.17
21.7
```

Explanation of the first output:

In the first example, there are 80 robots arriving per hour and 2 repair doctors working. Each repair doctor can treat 50 robots per hour. We start by calculating the arrival-to-treatment ratio.

$$\rho = \frac{\lambda}{c \cdot \mu}$$
$$\rho = \frac{80}{2 \cdot 50}$$
$$\rho = 0.8$$

Now that we have our ratio ($\rho = 0.8$), we can use it to look up in the dictionary the probability of having both repair doctors occupied simultaneously. This probability corresponds to $P_{full}(0.8,\ 2)$from the dictionary provided.

$$P_{full}(0.8,\ 2) = 0.71$$

We can now calculate the number of patients waiting per hour (L_q) and the total number of patients in the hospital per hour (L).

$$L_q = P_{full}(\rho,\ c) \cdot \frac{\rho}{1-\rho}$$
$$L_q = P_{full}(0.8,\ 2) \cdot \frac{0.8}{1-0.8}$$
$$L_q = 2.84$$

$$L = \frac{\lambda}{\mu} + L_q$$
$$L = \frac{80}{50} + 2.84$$
$$L = 4.44$$

Finally, what we want to find is the waiting time in minutes. To do this, we first calculate the waiting time in hours and then convert it into minutes.

$$W = \frac{L}{\lambda}$$
$$W = \frac{4.44}{80}$$
$$W = 0.555\ heures$$
$$W = 3.33\ minutes$$

Part 3: Heart monitor (40 points)

During the operations on the robots, to ensure the proper functioning of the electrical circuit, you want to implement a heart monitor. Your heart monitor will display the heartbeats made by the battery for 4 beats.

You will receive the systolic number, the diastolic number, and the heartbeat frequency to generate the visual of your heart monitor. We will create a simplified visual with these values, representing the electrical pulses of the robot's battery (Yes, your robot will use AC in this situation!). The systolic value will determine how high the peaks will go, while the diastolic value will determine how low the peaks will go. Finally, the heartbeat frequency will tell you how much space there is between each pulse.

Each increment of 20 will be an increase of one line above neutrality in the peak display for systolic, and each increment of 20 will be a decrease of one line below neutrality for diastolic. For example, a systolic value of 95 corresponds to an increase of 4 lines, and a systolic value of 100 corresponds to an increase of 5 lines from neutrality. A similar logic applies to the diastolic number; for instance, a diastolic number of 77 corresponds to a decrease of 3 lines, and a diastolic number of 85 corresponds to a decrease of 4 lines. In one beat, you will start at neutrality and rise by the number of lines corresponding to the systolic number, followed by a plateau at the top. Then, you will descend back to neutrality, another plateau, then descend according to the diastolic number, another plateau, and finally return to neutrality, which will be maintained according to the heart rate.

For the heart rate, each decrement of 10 below 180 will add one more line to the neutral value between two beats. So, a heart rate of 135 gives a neutral space of 4, and a heart rate of 142 gives a neutral space of 3 between beats.

You will need to display 4 heartbeats of the robot according to the systolic number, the diastolic number, and the heart rate.

Input and output specification:

For input, you will receive:

- systolic: the systolic as an *int*
- diastolic: the diastolic as an *int*
- heart_rate: the heart rate as an *int*

As an output, you'll need to return an *array of string* that is the visual of 4 heartbeats.

Sample input:

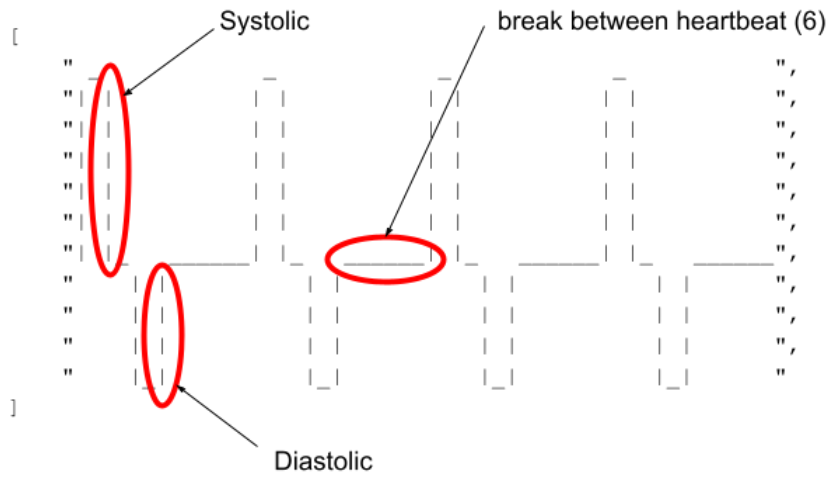
```
systolic: 120, diastolic: 90, heart_rate: 120
systolic: 105, diastolic: 87, heart_rate: 105
systolic: 100, diastolic: 92, heart_rate: 135
```

Sample output:

```
[
    "  _                 _                 _                 _                 ",
    " | |               | |               | |               | |                ",
    " | |               | |               | |               | |                ",
    " | |               | |               | |               | |                ",
    " | |               | |               | |               | |                ",
    " | |               | |               | |               | |                ",
    " | |_    _______   | |_    _______   | |_    _______   | |_    _______    ",
    "       | |               | |               | |               | |          ",
    "       | |               | |               | |               | |          ",
    "       | |               | |               | |               | |          ",
    "       |_|               |_|               |_|               |_|          "
]
[
    "  _                   _                   _                   _            ",
    " | |                 | |                 | |                 | |           ",
    " | |                 | |                 | |                 | |           ",
    " | |                 | |                 | |                 | |           ",
    " | |                 | |                 | |                 | |           ",
    " | |_    __________  | |_    __________  | |_    __________  | |_    ______",
    "       | |                 | |                 | |                 | |    ",
    "       | |                 | |                 | |                 | |    ",
    "       |_|                 |_|                 |_|                 |_|    "
]
[
    "  _             _               _               _              ",
    " | |           | |             | |             | |             ",
    " | |           | |             | |             | |             ",
    " | |           | |             | |             | |             ",
    " | |           | |             | |             | |             ",
    " | |_    _____ | |_    _____   | |_    _____   | |_    _____   ",
    "       | |             | |             | |             | |      ",
    "       | |             | |             | |             | |      ",
    "       | |             | |             | |             | |      ",
    "       |_|             |_|             |_|             |_|      "
]
```

Explanation of the first output:

In the first example, we have a systolic value of 120. Since we operate in increments of 20 on the display, 120 corresponds to 120 / 20, which equals 6. In the result, we can see that the beats have a height of 6, indicated in the image by the "Systolic" arrow. For the diastolic value, we have 90, which when divided by 20 gives 4.5, rounded down to 4. We can see that there is a movement of 4 lines downward in the "Diastolic" portion of the image. For the waiting time between beats, we take the heart rate, which is 120. We can do 180 - 120, which gives 60. This time, we use increments of 10. So, we have 60 / 10, which gives 6 neutral lines before the next beat. We repeat the shape of the beat for 4 beats to get the final result.



Part 4: Diagnostic Algorithm (40 points)

Between genetic predispositions, conditions already affecting the patient, side effects caused by medication, and so many other external factors such as the patient's environment, medical diagnosis is a pretty tough job, especially when it comes to rarer diseases. By getting rid of several more complex factors such as age and genetic predisposition, let's see if it's possible to build the basis of an algorithm that could diagnose a patient from a simple set of information and databases.

Your program will have access to two databases small enough to build a solid foundation. The first is called *diagnostics*, and contains the names and symptoms of the diseases you'll be diagnosing. The second is called *médicaments*, and contains a few common prescriptions that may concern your patients, as well as a list of the main side effects associated with each prescription.

For each patient, you'll see a list of their apparent symptoms, as well as a list of their prescriptions. **It is possible for a patient to be under no prescription at all, or to be under certain prescriptions that are absent from the database.** In the latter case, you should ignore the prescription.

The diagnostic logic you'll need to use is quite simple: we assume that a disease only shows symptoms that are present in its list in the database. **If a symptom is exhibited by the patient, but is not in the list given for a disease, we assume that it cannot reasonably be that disease.** Watch out! **Some symptoms** may be included in the list of **secondary effects produced by a patient's prescription**. As it cannot be proven whether these symptoms are due to the patient's condition or their prescription, **they must be ignored**. In this section, we assume that each patient will be affected **by one and only one disease** from the diagnostic database.

Input and output specification:

For input, you will receive:

- symptomes: a list of *string* containing the patient’s symptoms
- prescriptions: a list of *string* containing the patient’s prescriptions
- diagnostics: a 3D list of *string* corresponding to the diagnostic database
- médicaments: a 3D list of *string* corresponding to the prescriptions database

As an output, you’ll need to return a *string* corresponding to the patient’s diagnostic as it is written in the database.

Sample input:

```
symptomes: ["Fatigue", "Faiblesse", "Toux"], prescriptions:
["Ibuprofène", "Morphine"], diagnostics, médicaments

symptomes: ["Pâleur"], prescriptions: [], diagnostics, médicaments

symptomes: ["Troubles de vision", "Étourdissements", "Maux de tête",
"Perte de poids"], prescriptions: ["Acétaminophène"], diagnostics,
médicaments

symptomes: ["Douleurs abdominales", "Diarrhée", "Constipation",
"Fatigue", "Dépression", "Troubles d'équilibre", "Fièvre",
"Somnolence"], prescriptions: ["Antibiotiques", "Fluvoxamine"],
diagnostics, médicaments

symptomes: ["Éternuements", "Diarrhée", "Nausées", "Maux de gorge",
"Congestion nasale", "Fatigue", "Toux"], prescriptions:
["Antibiotiques", "Diphénhydramine"], diagnostics, médicaments
```

Sample output:

```
“Tuberculose”
“Anémie”
“Diabète”
“Sclérose en plaques”
“Rhume”
```

Explanation of the first output:

See the test template file for this part to see *diagnostics* and *médicaments* in full. The first example contains “Ibuprofène” and “Morphine” as prescriptions. Consulting the *médicaments* database, we find that there is nothing for “Morphine”, so it is ignored. For “Ibuprofène”, however, we find the following secondary effects: ["Démangeaisons", "Insomnie", "Nausées", "Vomissements", "Douleurs/Crampes d'estomac", "Diarrhée", "Fatigue", "Indigestion", "Constipation", "Ballonnements"]. We must then rule “Fatigue” out of the symptoms. With “Faiblesse” and “Toux” remaining, we find that the only condition capable of exhibiting both symptoms is “Tuberculose”: ["Toux", "Douleurs thoraciques", "Faiblesse", "Fatigue", "Perte de poids", "Perte d'appétit", "Frissons", "Fièvre", "Sueurs nocturnes"].

Part 4 Bonus

This case is rather more complex and therefore not worth any points. It is, however, very interesting for those who wish to push their Part 4 code a little further.

We now consider the case where the patient has symptoms that are not all present for a single disease in the database. In that case, we consider that the most likely scenario is that the patient suffers from two conditions in the database simultaneously. So, if a patient is affected by two conditions, any one of them can contribute to the list of symptoms, and a symptom therefore only has to appear in the list of at least one of the two conditions.

Here's an example of symptoms: [“Faiblesse”, “Toux”, “Fatigue”, “Écoulement nasal”, “Perte d’appétit”, “Maux de tête”, “Frissons”, “Fièvre”, ”Étourdissements”]. At first glance, it could be tuberculosis (“Tuberculose”), but you realize that the runny nose (“Écoulement nasal”), headache (“Maux de tête”) and dizziness (“Étourdissements”) are too much. You might think it's tuberculosis and another condition, but you realize that no condition combines these three missing symptoms. Looking for other possible combinations, we realize that flu (“Grippe”) symptoms include fatigue, cough, runny nose, fever, chills, loss of appetite and headache ([“Toux”, “Fatigue”, “Écoulement nasal”, “Perte d’appétit”, “Maux de tête”, “Frissons”, “Fièvre”]). Anemia (“Anémie”) has the remaining two symptoms, weakness and dizziness (“Faiblesse” et “Étourdissements”), plus other symptoms in common with the flu. In this case, it would be both flu and anemia. Your code's logic must be pretty robust to take such possibilities into account!