Lëtzebuerger Informatiksolympiad 2022

Finals

Task Descriptions

Instructions

- The allowed programming language is exclusively C/C++.
- All the programs must be realized in the form of a console application. For instructions how to realize a console application in the allowed programming languages, please refer to the remarks on the site *www.infosolympiad.lu* under the heading *Les questionnaires.*
- Under the input of the program is meant either the direct entry of data from the keyboard or the redirection from a text file in console mode. Under output of the program is meant either the direct display of data to the screen or the redirection to a text file in console mode.
- The formats of the input and output data shown in the execution examples must absolutely be respected.
- For testing, submitting and evaluating a program, the source file with a file extension "c/cpp" must be uploaded to the automated online judge CMS (Contest Management System), accessible via the homepage *www.infosolympiad.lu* or directly via the URL *http://158.64.46.20*. Please use your personal login (username & password) to access your account on the CMS. The filename of the single source file should be the same than the task name. Please refer to the CMS for technical details on how to test and submit a program.
- Please refer to the CMS for technical details like time limits and memory limits as well as compilation commands.
- You have the right to ask questions via the CMS, but the answers will not teach you how to use a programming language nor tell you how to solve the tasks by using a specific algorithm. The questions should be in relation with the CMS or should treat clarification issues concerning the task descriptions.

Any participant found to be in a situation of fraud during the finals will be excluded from the competition. Is considered as fraud any use of programs or portions of programs which would represent a plagiarism. However, documentation of the use and implementation of programming language instructions is permitted.

Task 1 menu 60 points

Description

Your friend Lio takes you to a fancy restaurant which has a menu consisting of **n** items, where the **i**-th item costs $\mathbf{c_i}$ euros. You also know if the item is food or drink. Last time she was here the food was great and so she brought you along this time. Looking at the menu again however, she realises that she forgot what she ordered last time. She thinks she remembers what the exact price **p** of her meal was and if she had something to drink. A meal always includes at least one food item. Please help Lio figure out what meal she had, if possible.

Task

Implement an algorithm that determines whether there is a selection of items on the menu which add up to the price **p** that Lio remembers and includes at least one drink if she remembers to have had a beverage with her meal. If she remembers not having drunk anything then the selection should not include any drinks.

Constraints

1 ≤ **n** ≤ 500

1 ≤ **p** ≤ 2000

1 ≤ $\mathbf{c_i}$ ≤ 50

Input and output of the program

Input data

The first line contains the integers **n**, **p** and either of the character strings “drink” or “nodrink”, separated by spaces.

The following **n** lines each contain an integer and a single character. Line **i** (for 1 ≤ **i** ≤ **n**) contains the integer $\mathbf{c_i}$ and either the character ‘f’ if it is a food item or ‘d’ if it is a drink instead.

Output data

Output ‘possible’ if it is possible to choose a meal according to Lio’s preferences or ‘impossible’ if it is not.

Execution examples

Example 1

Input file	Result

`5 23 drink`
`3 f`
`12 f`
`7 f`
`4 d`
`3 d`
	`possible`

It is possible that Lio had items 2, 3 and 4, which adds up to 12+7+4 = 23 and includes a drink.

Example 2

Input file	Result
5 23 nodrink 3 f 12 f 7 f 4 d 3 d	impossible

It is not possible to compose a menu without drinks that adds up to 23, since all the available items only add up to 22 euros.

Distribution of points

Subtask	Points	Description
1	15	Lio is certain to have eaten exactly one item of food as well as a single drink
2	15	Lio did not drink anything during her meal
3	30	No additional constraints

Task 2 escapegame 40 points

Beware

This is an interactive task, please read the assignment carefully, and respect the given template.

Description

You are invited to play a timed escape game, where you must find a secret object to win. The rooms are labelled from 1 to **N**. You can ask the game master if a certain set of rooms contains the secret or not, the only condition is that these room must be "easily accessible". That means that from each room from the given set you can access all others by only staying in this set of rooms (cf. Examples).

You know that the **N** rooms are connected and that the amount of corridors are equal to **N**-1.

Task

Write a code that finds in which room the secret is located.

Functions

Please use the given templates, where everything is already prepared for you.

You are to code a function ***int guess(int N, vector<int> u, vector<int> v)*** where **N** is the number of rooms, and where **u** and **v** represent the corridors, i.e. rooms number **u**[**i**] and **v**[**i**] have a corridor between them. It should return where the secret object is located. These vectors are of size **N**-1.

You can call the function ***query(vector<int> rooms)*** that returns a **bool** telling you if the secret is in one of the given rooms. ***rooms*** containing the labels of the rooms you want to have checked. The subset must be "easily-accessible", else your code will be terminated for **Protocol Violation**.

Example

Let's use the call example here.

The sets of rooms {1, 2, 3} and {1, 2, 4, 5} are examples of "easily accessible" rooms, since you can reach any room from any room by staying in the selected rooms.

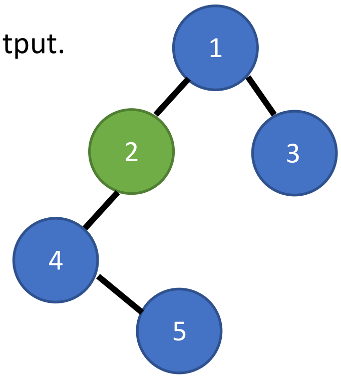
The set {1, 2, 3, 5} is **not** easily accessible, since you have to go through room 4 to access any room starting from room 5.

Constraint

N ≤ 512

Input and output of the program

You are not responsible of reading or writing anything from the standard input or output.



Call example

The function is called with the arguments **N** = 5, **u** = {1, 1, 2, 4} and **v** = {2, 3, 4, 5}.

Let's say the secret is in room 2.

The call "guess({1,2,3})" will return true, since it contains room 2

The call "guess({1,3})" will return false, since it does not contain room 2

The call “guess({1,2,3,5})” will halt the program, since the given set of

rooms is not “easily accessible”.

Your function is to return 2 (in this example) given the information found using the calls to guess.

Input format

The input format, if you choose to create your own testcases, should respect the following format:

The first line should contain **N** and the label of the secret’s location.

The next **N**-1 lines contain **u[i] v[i]**

Input of the call example
`5 2`
`1 2`
`1 3`
`2 4`
`4 5`

Distribution of points

Let **Q** be the highest number of guesses along all testcases of a certain subtask.

If any testcase failed, you won’t get any points for the subtask.

Subtask	Points	Description	Point repartition
1	10	**N** ≤ 16	Points = 10*exp(-0.2(**Q**-4)): If **Q** ≤ 4, you’ll get all 10 points, if **Q** = 16, you’ll get 1 point
2	30	No additional constraints	Points = 30*exp(-0.0046(**Q**-9)): If **Q** ≤ 9, you’ll get all 30 points, if **Q** = 512, you’ll get 3 points

Remark

The given grader files do not test if the given set of rooms really is easily accessible. The grader on the Contest Management System does.