Fall 2022 UMW Programming Contest

Overview

Welcome to the Fall 2022 ACM Programming Competition! This is a contest wherein you compete amongst your fellow students, and professors, in an effort to solve a set of programming problems as fast as possible.

Problems

There are ten problems of (roughly) increasing difficulty. The team who solves the most problems is declared the winner. In the event of a tie, the team with the best time wins. The best time also includes penalties for wrong submissions. Each problem has a general description, details on the formats for input and output, and example input and output.

Input is done from standard input and output is done to standard output. *Do not open any files inside of your program.* Output of your program must match that of the correct output *exactly*. You can solve the problems in C, C++, Java, or Python. Up to date versions of these languages will be used to judge your entries. You should save your code using the standard extension for your language of choice (.c .cpp .java and .py respectively).

Rules

You are allowed to have any printed material such as books or printouts. You are also allowed to access standard library documentation for your language of choice. *No other use of the internet is permitted.* You are also not allowed to copy and paste code from any source be it the web, a thumb drive, or another file that you have. Only one computer is allowed to a team.

Logging In

To log in to the system, direct your browser to `https://bit.ly/3TKxH7w`. You will then need to enter your team name and password which are provided for you. This will bring up the interface for you to submit solutions to the problems, ask questions (which will be broadcast to all participants), and see how your team is doing.

Submitting

To submit a solution, first make sure that you select the correct problem letter on the top of the site. Then click the "Choose File" button. Then choose the file that contains your source code (be sure it has the correct extension). After that it will be submitted for judging. It will receive an automatic judging response which is marked pending. Your solution will then be checked manually and a final response will be assigned.

DO NOT TURN OVER THIS SHEET UNTIL THE CONTEST STARTS

Problem A: Say My Name

Most Pokemon only communicate by saying their own name with different inflections. They often repeat their name many times to form a “sentence”. Write a program to read in a Pokemon’s name, and a number, then output the Pokemon’s name that many times.

Input

Input will consist of two lines. The first line is the name of the Pokemon. The second line is a number indicating how many times to repeat the name.

Output

Output should consist of a single line containing the name of the Pokemon repeated, separated by spaces.

Sample Input

```
Pikachu
7
```

Sample Output

```
Pikachu Pikachu Pikachu Pikachu Pikachu Pikachu Pikachu
```

Problem B: Making Change

Poke Marts sell many items that are helpful for Pokemon trainers including poke balls, potions, Pokemon repellents and escape ropes. You work in a Poke Mart, and your boss has asked you to write a program to calculate the change given to trainers after they make a purchase.

Your program will be given a list of integer values representing the cost in Pokedollars for the items the trainer is buying. It will also be given the amount of Pokedollars the trainer pays with. Your program will then compute the change returned to the trainer, or inform them they do not have enough money.

Input

The first line of input will be an integer giving the number of items the trainer is buying. Next, there will be that many lines giving the price of each item. The last line of input will be the amount the trainer gives the cashier.

Output

Output will be of the form "Change is <amount>." where <amount> is replaced with the amount of change the trainer is due, or "Not enough money!" if the trainer is short.

Sample Input 1

```
4
200
700
450
200
1800
```

Sample Output 1

```
Change is 250.
```

Sample Input 2

```
4
200
700
450
200
1200
```

Sample Output 2

```
Not enough money!
```

Problem C: Nidoran Explosion

Reggie is a Pokemon breeder who specializes in breeding Nidorans. Reggie starts with two Nidorans that are just born. Every pair of Nidorans that Reggie has that begins a year at 3 years of age or more is able to produce an offspring Nidoran each year.

Reggie wants to be able to predict how many Nidorans he will have after a given number of years. He has asked you to write a program that will take in a number, N, and compute how many Nidorans there will be after N years. You can assume that no Nidorans will die.

Input

Input will be a single integer, the number of years.

Output

Output should be of the form “There will be <number> Nidorans after <N> years.”

Sample Input

```
17
```

Sample Output

```
There will be 51 Nidorans after 17 years.
```

Problem D: Can you Dig It?

The Pewter Museum of Science in Kanto is holding a treasure hunt for young Pokemon trainers. The hunt consists of an 8x8 plot of land. In one of the sections, is buried a Pokemon fossil, a rare prize.

In each of the other 63 sections, is a piece of paper with a 2-digit number written on it. This number is a clue directing the trainer to the next section of land to search. The tens digit indicates the row number, and the ones digit indicates the column number.

Participants in the treasure hunt start in the first section (row 1, column 1) and follow the clues until they either find the fossil, or go back to a section of land they have already searched. Your goal is to write a program which, given what's buried in each section of land, will determine whether or not the participants will find the fossil.

Input

Input will consist of 8 lines. Each line corresponds to a row in the plot of land, and contains 8 entries. Each entry is either a two-digit number indicating the cell of the next clue, or the value 0 which indicates that the cell contains the fossil.

You can assume that no clue will refer to a non-existent cell.

Output

Your program should output output the cell numbers that are searched at each step. If the participant finds the fossil, you should end by printing "Found the fossil!". If the participant is sent back to search a section that they have already searched, you should print "Fossil not found!".

Sample Input

```
31 21 66 12 52 87 64 44
11 13 78 17 44 51 36 33
76 44 12 54 84 71 12 33
24 53 15 68 58 0 14 88
22 44 61 11 87 0 28 64
88 22 43 43 54 17 42 81
56 23 0 11 17 24 56 71
61 19 21 82 39 0 77 27
```

Sample Output

```
11
31
76
24
17
64
43
15
52
44
68
81
61
88
27
36
71
56
Found the fossil!
```

Problem E: Broken Messages

Team Rocket wants to send messages amongst their operatives without local authorities or meddlesome trainers being able to read them. To communicate safely, Team Rocket uses a simple substitution cipher for their messages. This works by mapping one letter onto another. For example, the cipher might look like this:

A	B	C	D	E	F	G	H	I	J	K	L	M
V	H	P	L	Q	F	T	C	Y	N	U	G	X

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	I	Z	E	M	K	W	A	J	R	D	O	S

Then, when Team Rocket wants to send a message, they replace each letter with the one it is mapped to. For the example cipher above, the word “cafe” would be encoded as “pvfq”.

When an operative receives a message they use the same substitution scheme to decode the message. (Each Team Rocket operative is issued a book with all of the different substitutions and when to use them).

Unfortunately, many Team Rocket operatives are not very careful, and make mistakes when doing the substitution. This can lead to all kinds of trouble.

Your goal is to read in the original message, and the encoded message, and check for potential mistakes the operative may have made in the substitution. To do this, you will look for the following potential errors:

1. Two identical characters in the original message being encoded differently. For example, in “weedle” -> “fqrbxp”, the two e’s have different substitutions indicating an error.
2. Two identical characters in the encoded message corresponding to different characters in the original message. For example, in “froakie” -> “ppciwkl”, f and r are both encoded as p, also indicating an error.

Of course, it would be better for Team Rocket to use a computer program to do the substitution in the first place, but Team Rocket did not get where it is today by making good decisions.

Input

Input will consist of two lines of lowercase letters, spaces, and punctuation. The first line is the original message, while the second is the encoded message.

Spaces and punctuation are not encoded, and you can assume there are no errors in these characters.

Output

If you detect an error, your program should output "Error detected." Otherwise, it should output "No error detected."

Sample Input

```
we will steal pikachu at midnight!
av azoo hqvfo emufykt fq dzmszpkq!
```

Sample Output

```
Error detected.
```

Problem F: I Choose Who?

During a Pokemon battle, trainers try to send out Pokemon who will beat their opponent's Pokemon. Which Pokemon to send against another can be a difficult decision. Trainers primarily base this on the level and type of the Pokemon involved.

A level is a number in the range of 1 – 100 indicating how strong and experienced the Pokemon is.

There are different types of Pokemon, and some types have advantages in battle over other types. The following table lists seven common types of Pokemon along with estimates on how strong they are against other types.

	Normal	Fire	Water	Grass	Electric	Fighting	Psychic
Normal						-5	-5
Fire			÷2	×2	+10		
Water		×2		÷2	÷2	+5	
Grass		÷2	×2				
Electric		-10	×2			÷2	
Fighting	+5		-5		×2		÷3
Psychic	+5					×3	

The rows indicate the type of your Pokemon, while the columns indicate the type of your opponent's. The entries in the table indicate a modifier on the levels of the Pokemon involved.

For example, if you have a Water type Pokemon, while your opponent has a Fire type, you will have a "×2" modifier. So, for example, a level 10 Water Pokemon would be equivalent to a level 20 Fire Pokemon. Likewise, a level 35 Psychic Pokemon would be equivalent to a level 40 Normal Pokemon.

If there is no entry in the table, it means the corresponding Pokemon have no particular advantage or disadvantage against each other.

Your goal is to write a program to help trainers choose which Pokemon to send into battle based on the level and type of the opponent, and the level and type of the Pokemon the trainer has available to them.

The strategy your program should take is to send out a Pokemon with a type-adjusted level that is greater than the opponent. If there are multiple such Pokemon, you should send the one with the lowest actual level (to keep your strongest Pokemon as long as possible). If there

are no Pokemon with a type-adjusted level higher than the opponent, you should send the Pokemon with the lowest actual level (to keep your strongest Pokemon as long as possible).

Input

Input will consist of a list of 7 Pokemon. The first Pokemon given will be the opponent. The remaining 6 Pokemon will be the ones in the trainer's party that you have to choose from.

Each Pokemon will have a name, level, and type.

Output

Output will consist of the phrase "<Name>, I choose you!", where <Name> is replaced with the name of the Pokemon being chosen.

Sample Input

```
Growlithe Fire 27
Meowth Normal 30
Magikarp Water 3
Pidgey Normal 14
Quilladin Grass 52
Squirtle Water 15
Electrike Electric 24
```

Sample Output

```
Squirtle, I choose you!
```

Problem G: Safari Zone Planning

The Safari Zone in Hoenn is a reserve for rare Pokemon. Each type of Pokemon has their own area in the Safari Zone to live. Pokemon trainers can come to the Safari zone to see Pokemon they would otherwise not encounter.

The Safari Zone is being renovated and the Hoenn city council is considering how to divide the space into areas for each type of Pokemon. One thing that is essential is ensuring that the areas allocated for each type of Pokemon do not overlap. This is especially important since some Pokemon in the Safari Zone, such as Scyther and Electabuzz, are quite hostile to each other.

The Hoenn city council has asked you to write a program to analyze the proposed area divisions and ensure they do not overlap.

Each area is represented by the x and y coordinates of the lower-left, and upper-right corners. The coordinates will be integer numbers in the range of (0 - 100). The point (0, 0) is in the lower-left hand corner of the Safari Zone. Each area will be rectangular and aligned horizontally and vertically.

Input

The first line of input will be an integer giving the number of areas. Each subsequent line of input corresponds to one area. Each of these lines will have four integers: x1 y1 x2 y2 where (x1, y1) are the coordinates of the lower-left corner of the area, and (x2, y2) are the coordinates of the upper-right corner of the area.

Output

Your program should output either “Areas overlap.” or “Areas do not overlap.”

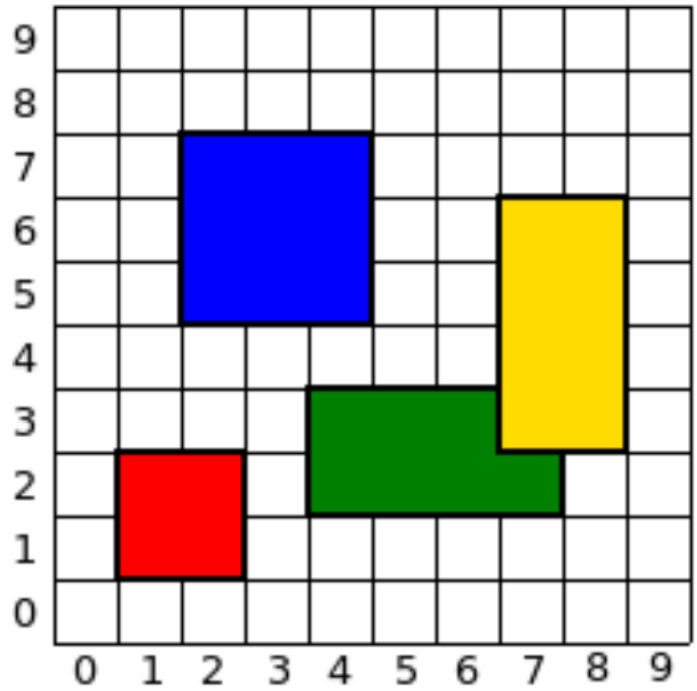
Sample Input

```
4
1 1 2 2
2 5 5 8
4 2 8 4
7 3 9 7
```

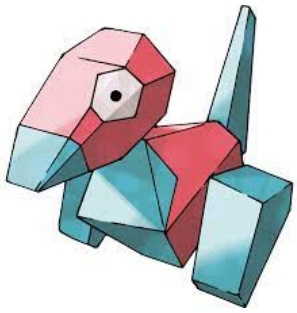
Sample Output

```
Areas overlap.
```

Visualization of Sample Input



Problem H: Porygon



Porygon is a man-made Pokemon that was created in the Pokemon Lab in the Kanto region. It is made up entirely of computer code representing data and commands. Porygon can move freely through cyberspace.

Porygon's data memory consists of an array of one thousand integers. Porygon also has a memory index that indicates which memory cell is being operated on.

Porygon is capable of the following attacks:

Number	**Attack**
1	Tackle
2	Conversion
3	Sharpen
4	Psybeam

Rather than have the trainer choose which attacks Porygon uses, they are decided by a control program. This program consists of the following commands:

Command	**Meaning**
>	Increment the memory index by 1.
<	Decrement the memory index by 1.
+	Increment the integer stored at the memory index by one.
-	Decrement the integer stored at the memory index by one.
!	Use the attack corresponding to the integer stored at the memory index (1, 2, 3 or 4).
[	If the integer stored at the memory index is equal to zero, then jump to the instruction after the matching "]" command.
]	Instead of executing the next sequential command, jump to the matching "[" command.

For each command besides [and], Porygon moves on to the next command after executing the current one. If there is no next command, the program halts (i.e. when the last command finishes). Porygon matches brackets using the same method by which parentheses are typically matched.

Each of Porygon's one thousand memory cells is initialized to 0. Porygon's memory index is also initialized to 0. Porygon begins executing the commands starting with the first one.

Your job is to write a program to help trainers decipher Porygon control programs. You will read in Porygon commands, simulate them as Porygon would, and output the sequence of attacks Porygon will use.

You can assume that the control program has properly matching bracket commands, and will terminate. You can also assume that the ! command will only be used when the integer stored at the memory index is in the range 1-4. Lastly, you can assume that there will be fewer than 1,000 commands in the input program.

Input

Input consists of a single line consisting of a sequence of Porygon commands.

Output

Output consists of a list of the attacks the control program takes, one per line.

Sample Input

```
+++++>+>++>+++>++++<<<<[>!>!<<-]>>>>!
```

Sample Output

```
Tackle
Conversion
Tackle
Conversion
Tackle
Conversion
Tackle
Conversion
Tackle
Conversion
Psybeam
```

Problem I: Finding the Safest Path

Wally, a fellow trainer who is younger than you has a problem. He often has all of his Pokemon in poor health and needs to find his way to the Pokemon center. The Pokemon center provides healing to Pokemon who have fainted or are low on health. The problem is, there are often areas with tall grass on the way to the Pokemon center.

Wild Pokemon hide in tall grass and attack trainers as they walk through. Additionally, many wild Pokemon like to rest near trees. While walking in tall grass near a tree, Wally is twice as likely to be attacked by a wild Pokemon. Wally wants you to help him find a way to the Pokemon Center minimizing his chances of being attacked by wild Pokemon.

Wally can only move in the four cardinal directions.

Input

Input consists of a 10x10 2D grid of characters representing the area Wally must travel. The following characters are used for the following meanings:

Character	Meaning
(space)	A path.
`^`	Tall grass.
`T`	A tree (which Wally cannot walk over).
`W`	Wally's starting position.
`P`	The Pokemon center.
`X`	An impassable space (such as water or a building).

Output

Output should be a numerical score representing Wally's minimal chance of being attacked by a wild Pokemon. For each space of tall grass Wally must walk through, add one to the score.

If the space of tall grass is next to a tree (in any of four cardinal directions), an additional one is added to the score.

If the Pokemon center is completely blocked by trees or impassable spaces,print "Impossible to get to the Pokemon Center!".

Sample Input

```
^^^^^^^^^^
^P  ^^T^^^
^^^^^^^^^^
^^T^^^^^^^
XX^^^^ ^^^
XX^^^^ ^^^
^^^    ^^^
^^^ ^^^^^^
^^^ ^^^^^^
^W  ^^^T^^
^^^^^^^^^^
```

Sample Output

```
Score is 5.
```

Problem J: Is Better Bigger?

Professor Rowan has a theory about Pokemon: he believes that the experience level of Pokemon is correlated with the size of the Pokemon. That is, for Pokemon of the same type those at higher levels will be larger than those of lower levels.

He is testing this theory using the Pokemon Snorlax as an example. He has found a number of Snorlax throughout the region and recorded both their level, and their weight. He believes that they should be correlated, but is looking for counter-examples to his theory.

He has asked you to go over the Snorlax samples and find the largest sequence that can be made where the levels increase, but the weights decrease. He won't count sequences in which either the levels or the weights are the same.

For example, let's say the sample contains a Snorlax who is level 11 and weighs 402 kilograms. And another is level 12 and weighs just 395 kilograms. These two form a sequence of two that goes against Professor Rowan's theory.

Your job is to find the largest possible subset where levels strictly increase but weights decrease.

Input

The first line of input is a number N giving the number of Snorlax in the data set.

Following that are N lines of input. Each of these lines lists the level of the Snorlax, followed by a space, followed by the weight of the Snorlax in kilograms.

It's possible that multiple Snorlax will have the same level, the same weight, or both.

Output

The first line of output should say "There are M Snorlax in the sequence:" where M is the size of the largest sequence such that each successive Snorlax has a greater level and a smaller weight.

Following that are M lines of output which indicate the Snorlax that comprise the sequence. These lines contain a single integer between 1 and N inclusive which correspond to where that Snorlax appeared in the input.

Sample Input

```
9
14 375
9 404
7 417
14 380
16 302
11 370
3 375
14 275
17 320
```

Sample Output

```
There are 4 snorlaxes in the sequence:
3
2
6
8
```