Contents

1 Counting Swann's Coins

Governor Weatherby Swann orders you to count the number of coins in the government treasury. To make the job more interesting you decide to say "Dead" for all numbers that are a multiple of 3 and "Man" for all numbers that are a multiple of 5. For numbers that are multiples of both 3 and 5 you say "DeadMan".

Input/Output Format:

The input will be the number of coins you need to count. The output will be the numbers and words in sequence, separated by spaces. Start a new line after each word.

Example:

Sample Input 1:

```
4
```

Sample Output 1:

```
1 2 Dead
4
```

Sample Input 2:

```
18
```

Sample Output 2:

```
1 2 Dead
4 Man
Dead
7 8 Dead
Man
11 Dead
13 14 DeadMan
16 17 Dead
```

2 Dividing the Pirate Hoard

After raiding an island N pirates end up with M coins. After the raid everybody gathers coins and puts them together in a community pile, to be divided the next day. During the night one pirate decides to take his share himself. He sneaks his way to the hoard and divides the M coins into N equal piles, with K coins left over. He keeps the extra K coins for himself, hides his pile of coins, and puts the other piles of coins back into a single pile. The other pirates then do the same thing one at a time, each dividing the remaining coins into N piles, taking one pile, and keeping any extra coins to keep the remaining piles even.

Given the number of pirates and coins, find out how many coins each pirate will have his own hidden pile (his equal-sized pile plus remaining coins) and how many coins are left in the community pile at the end of the night.

Input/Output Format:

The input will be the number of coins, followed by the number of pirates. The output should be the number of coins received by each pirate from largest to smallest (separated by spaces), followed by the total number of coins remaining in the hoard on a second line.

Example:

Sample Input 1:

```
12 2
```

Sample Output 1:

```
6 3
3
```

Sample Input 2:

```
10 3
```

Sample Output 2:

```
4 2 2
2
```

3 Pirates On Parade

Davy Jones wants his men to line up on parade. The men are to line up in pairs, from shortest to tallest. However, two pirates in the same pair can only differ in height by at most 2 inches. You start lining up the pirates in pairs in order, starting with the shortest pirate. If after selecting a pirate it is not possible to find a pirate sufficiently near his height (or if there are no pirates left), the unpaired pirate will be forced to walk the plank! You may assume no pirates are exactly the same height.

Input/Output Format:

The input will be a list of pirates listed by name and their height in inches as an integer. The output should be the names of pairs of pirates, each on a single line, in order from shortest to tallest. The shorter pirate in each pair should be first. Any pirates forced to walk the plank should not be listed.

Example:

Sample Input 1:
Sam 60 Dan 52 Bob 54 Art 61
Sample Output 1:
Dan Bob Sam Art

Sample Input 2:
Sam 60 Dan 52 Bob 54 Kim 70 Art 61
Sample Output 2:
Dan Bob Sam Art

4 Pirates' Code

Davy Jones has captured another ship and is smiling contently under the sun. Today is a good day for him. He will get more souls to serve on his crew. The day, however, looks so nice, the sun shining brightly above all and the ocean spilled calmly around, that he decides to be merciful and give some chance to the wretched seamen of the captured ship. He decides to play the following game.

He will line a group of captives in front of him, and then write a number of his choice on each man's forehead. After that, he wants to check if the set of numbers is 3-free. What does it mean for a set to be 3-free? It means that there are no 3 numbers in the set that form an increasing arithmetic progression (weird games have damned Jones, aye), i.e., a sequence of numbers such that the difference of any two successive members of the sequence is a positive constant, such as {1 2 3}, {4 6 8}, or {-2, 1, 4}. If the the set of numbers on men's foreheads is 3-free, the group is free, too (what an irony). However, if not, those who have the numbers of the lexicographically first triple that proves (i.e. is a witness) that the set is not 3-free will serve on Jones' crew for an eternity.

And you ... You will be Jones' assistant in this game. Checking each group whether it is 3-free or not. And you'd better check right or you will end up on Jones' crew as well.

A triple (a_1, a_2, a_3) is an increasing arithmetic progression if $a_2 - a_1 = a_3 - a_2$, and $a_2 - a_1 > 0$.

A triple (a_1, a_2, a_3) is lexicographically smaller than (b_1, b_2, b_3) if

- $a_1 < b_1$, or
- $a_1 = b_1$ and $a_2 < b_2$, or
- $a_1 = b_1, a_2 = b_2$ and $a_3 < b_3$.

Note that you will be looking at triples (a_1, a_2, a_3) of increasing order, i.e. $a_1 \leq a_2 \leq a_3$. The numbers on men's foreheads need not be in increasing order though.

Input/Output Format:

Input consists of a single line with the numbers written on the captured men's foreheads. The first number of the line denotes how many men are there in the group and should not be included in the sequence.

Output should consist of a single line. If the sequence of numbers in the input is 3-free, output "Sequence is 3-free." If the sequence is not 3-free, output "Sequence is not 3-free. Witness: w_1, w_2, w_2." (note the intervals after the first dot and the colon), where w_1, w_2, w_3 is the lexicographically first witness that the sequence is not 3-free.

Example:

Sample Input 1:
4 1 5 6 8
Sample Output 1:
Sequence is 3-free.

Sample Input 2:
7 1 3 5 2 -7 0 -1
Sample Output 2:
Sequence is not 3-free. Witness: -7,-1,5.

5 Protect Our Treasure!

A group of N pirates put their treasure in a chest. The pirates don't trust each other (with good reason!) so they consult a locksmith. The locksmith puts L locks on the chest in a way that every lock must be unlocked to open the chest. Then he distributes keys to the locks to the pirates so that every pirate has some but not all of the keys. Any given lock can have multiple keys, but any given key can only open one lock.

Given a number of pirates and sets of keys assigned to each pirate, you must find all groups of pirates that together can open the chest, but do not contain any unnecessary members (i.e., the chest cannot be opened if any pirate is left out of the group).

Note that there will be too many pirates for you to simply examine all possible combinations of pirates, so you will need to be more intelligent in your choice of algorithms.

Input/Output Format:

The first line of input will be the number of pirates N, followed by the total number of locks L. Following will be a line for each pirate (1..N) listing the number(s) of the locks (1..L) to which the pirate has keys. The output will be all possible combinations of pirates that can open the chest. Each combination will be on a separate line, with the pirates listed in order from smallest to largest. The combinations must be listed in order of smallest number of pirates to largest required. For combinations with the same number of pirates, list the combinations in lexicographic order (i.e., compare the 1st pirates, breaking ties by comparing the 2nd pirates, then the 3rd, etc...).

Example:

Sample Input 1:

```
3 3        // 3 pirates, 3 keys
1 3        // pirate 1 has keys to lock 1 and 3
1 2        // pirate 2 has keys to lock 1 and 2
2 3        // pirate 3 has keys to lock 2 and 3
```

Sample Output 1:

```
1 2
1 3
2 3
```

Sample Input 2:

```
4 3        // 4 pirates, 3 keys
1          // pirate 1 has key to lock 1
2          // pirate 2 has key to lock 2
3          // pirate 3 has key to lock 3
1 2        // pirate 4 has keys to lock 1 and 2
```

Sample Output 2:

```
3 4
1 2 3
```

6 Pirates' Path

Our captain Jack Sparrow is again on the natives' island where again the natives think he is a God. But Captain Jack ain't giving in easily (especially when his life is at stake) and is trying to escape. The island is divided into different areas separated from one another by the many rivers abundant on the island. Two areas are adjacent if they are connected through a bridge. Of course captain Jack has to make it from the area where he's been kept in captivity to the area where the Black Pearl, his dear ship, is located. Quite unfortunately, the natives are very eager to keep their "God" on the island so they have decided that they will guard some selection of the bridges by putting one man on each selected bridge. Now, our captain, being as unwilling as he is to spend any extra effort whatsoever on a job (as well as desirous of being merciful to his devotees), would like to incapacitate as few natives as possible on his way back. WARNING: Jack needs to make his decision QUICKLY!

Input/Output Format:

Input consists of $b + 1$ lines where b is the number of bridges on the island. The first line contains 4 numbers separated by a single space:

$n\ b\ s\ e$

n is the number of areas on the island, b is the number of bridges, s (where $0 \leq s < n$) is the area where Captain Jack is being held and e (where $0 \leq e < n$) is the area where the Black Pearl is located.

Each of the next b lines describes a bridge. The line will contain 3 numbers separated by a single space:

$a\ b\ c$

a and $b, 0 \leq a, b < n$ are the areas connected by the bridge described by the line and c is a bit, either 0 or 1, indicating whether the bridge is being guarded by a man or not. If it is being guarded, only one man will be a guard.

Note that there may not be a path from the area where Captain Jack is being kept to the area of the Black Pearl. If this is the case, output should be a single line containing the following (quotes for clarity):

"It's over with Captain Jack. At least till Pirates of the Caribbean 3."

In case there is a path to the Black Pearl, output should be a single line containing (quotes for clarity):

"x native(s) on the easiest way for Captain Jack."

where x should be replaced by the least number of natives guarding bridges on a path from the starting area to the area of the Black Pearl.

Example:

Sample Input 1:

```
7 11 0 6
0 1 0
0 2 1
1 2 0
1 3 1
2 4 0
1 5 1
3 5 0
3 4 1
3 6 0
4 5 1
5 6 1
```

Sample Output 1:

```
1 native(s) on the easiest way for Captain Jack.
```

Sample Input 2:

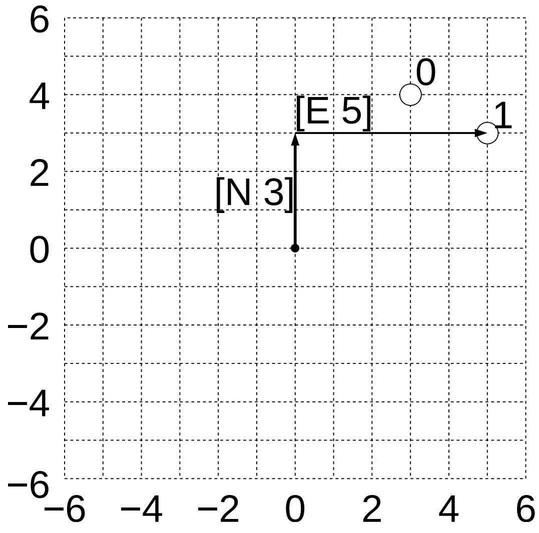
```
6 5 1 5
0 1 0
0 2 0
1 2 0
1 3 1
3 4 0
```

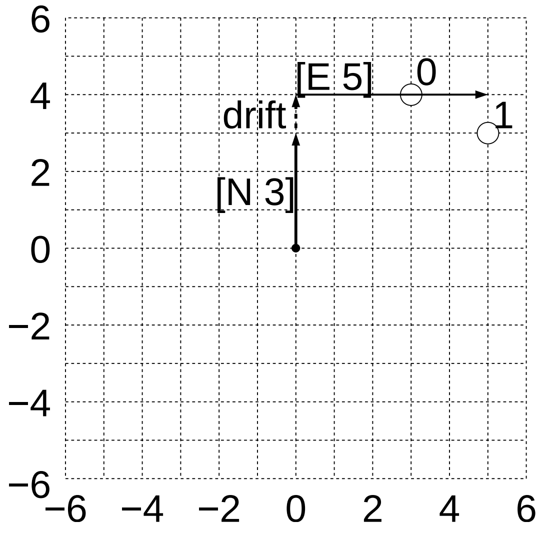
Sample Output 2:

```
It's over with Captain Jack. At least till Pirates of the Caribbean 3.
```

7 Navigating the Reefs

Captain Jack Sparrow is planning a cruise around the treacherous coral reefs of the Caribbean. He knows the locations of the reefs, and he has his route planned out. His route starts at coordinate $(0, 0)$. The route consists of a number of straight-line segments, each of which is given by one of the four *compass directions* (N, E, S, W) and a distance in miles. For example, [N 3] means that he travels north for 3 miles (placing him at $(0, 3)$). Suppose that the next day the segment is [E 5], which means that he then continues east for 5 miles (placing him now at location $(5, 3)$). If there is a reef at location $(5, 3)$, then his ship will collide with it and sink. (And poor Jack and his crew pay a visit to Davy Jones' locker. Arrrgh!)





There is an additional catch, however. At the end of each segment (including the last one) the ship may drift one mile in any one of the four compass directions, or it may not drift at all. The direction of the drift is entirely unknown, and it may be different for each segment. Jack must determine whether the voyage is free from any possibility of collision, no matter what combination of drifts occur. Let us consider the previous example, [N 3], [E 5]. Suppose that there is also a reef at location $(3, 4)$. This voyage is *not safe*, because after the first segment (which ended at $(0, 3)$, the ship might drift one mile to the north (placing it at $(0, 4)$), after which the next east segment will lead it straight into the reef. (And again, Arrrgh!)

Your job is to write a function that Jack can use to determine *whether the voyage is safe* from reefs. The function will be given the reef locations, followed by a series of segments. All quantities are integers. If there is any possibility of collision, your function will output the index of the *first* reef that could be hit. If the route is safe, then your function will return -1.

Input and Output

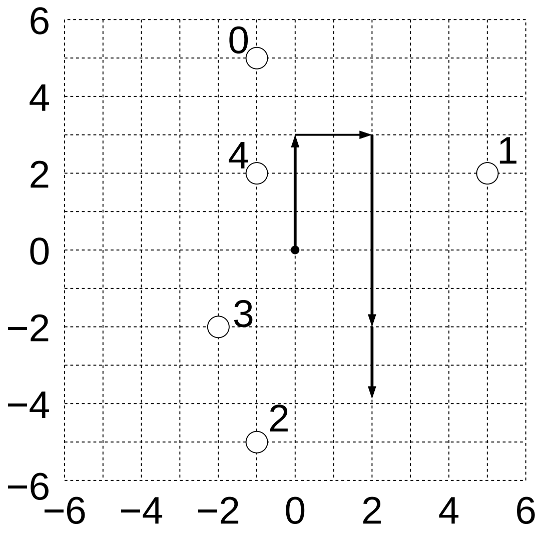
We have provided a main program for you that handles all the input and output. The quantities it inputs are listed below and are stored as global variables. The variable oceanLimit indicates the limits of the entire ocean. An input of 20, for example, means that the ocean area in which Jack's ship can travel is limited to $-20 \leq x \leq +20$ and $-20 \leq y \leq +20$. You may assume that Jack's voyage, even allowing for drifting, will never extend outside these limits. The variables nReefs and nSegs give the number of reefs and number of segments. The arrays reefX[] and reefY[] contain the x- and y-coordinates of the reefs, respectively. The array segDirec[] contains the direction of each segment, as a single character, and the array segLength[] contains the length of each segment in miles.

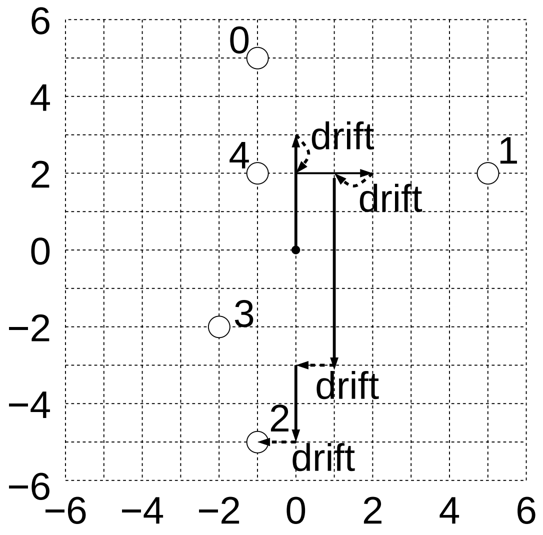
```
static int    oceanLimit;     // ocean min/max limits
static int    nReefs;         // number of reefs
static int    nSegs;          // number of segments
static int[]  reefX;          // reef x-coordinates
static int[]  reefY;          // reef y-coordinates
static char[] segDirec;       // segment directions ('N', 'S', 'E', 'W')
static int[]  segLength;      // segment lengths
```

All you need to do is to provide the body for the function `checkVoyage()`, which determines whether the voyage is safe from any possible collision. If so, it returns -1. If not, it returns the index r, where $0 \leq r \leq (\mathsf{nReefs} - 1)$, and r is the first reef that may cause a collision. The prototype is

```
private static int checkVoyage();
```

Consider the example below, in which the segments are [N 3], [E 2], [S 5], [S 2]. On the left we show the voyage without drifting. On the right we show the combination of drifts after each segment that result in a collision with reef 2. (Note that the ocean limit is ±20, rather than ±6 as drawn in the figure.)





Here is a sample of the program's input and output. In this case the function checkVoyage() returned the index 2.

Input:	Output (as generated by program):
20 5 -1 5 5 2 -1 -5 -2 -2 -1 2 4 N 3 E 2 S 5 S 2	Ocean limits: -20..20 Reefs: (-1,5) (5,2) (-1,-5) (-2,-2) (-1,2) Segments: [N 3] [E 2] [S 5] [S 2] Collision with reef 2 at (-1, -5)

8 Pirates' Gold

You have stolen the pirates' gold! You have a number of pieces of gold worth various amounts of money. You can think of your loot as a set of positive integers (with possible duplicates). For example, the set $\{7, 10, 13, 4, 10, 24\}$ means that you have:

- 1 piece of gold worth \$4.
- 1 piece of gold worth \$7.
- 2 pieces of gold each worth \$10.
- 1 piece of gold worth \$13.
- 1 piece of gold worth \$24.

(Note that the order in which the amounts are listed does not matter.)

The pirates catch up to you and demand that you give it back or you will have to "walk the plank." Luckily for you, pirates are not very good at math, and they do not know exactly how much you stole. They boldly say, "We claim that you stole at least \$$xxx$ dollars worth of gold!" Since you want to live, you need to give back *at least* as much as they claim. However, since you are greedy you want to give back *as little* additional as possible.

Write a program that will save your life, but save as much loot for you as possible. Given a list of the values of the gold that you stole and given how much the pirates claim you stole (both in dollars) determine the minimal amount that you need to give back so that you give back at least as much as you stole. If the total amount that the pirates claim exceeds the total that you stole, then you're dead! In this case your program should return the special value -1.

For example, suppose you have the stolen amounts $\{7, 10, 13, 4, 10, 24\}$ and the pirates claim that you stole at least \$25, then you would reason that by giving them $10 + 13 + 4 = 27$ dollars, your life will be saved. Furthermore, this is best for you, since there is no other way to form a smaller sum that is at least \$25.

Input and Output

We have provided a main program for you that handles all the input and output. (The first number is the pirates' claim followed by the amounts you stole.) The list of stolen amounts is stored in an array stolen[]. (All amounts are positive. You may obtain the number of values as stolen.length.) The pirates' claim of the amount you stole is given by the variable claim. (The claim a nonnegative integer.) You are to write the function payBack(), which will return the minimum amount you should pay back, or else return -1 if it is impossible to satisfy the pirates' claim.

```
    private static int payBack(int stolen[], int claim);
```

Here are some samples. In the first case payBack() returns 27 and in the other it returns -1.

Input:	Output (generated by our program):
`25     7 10 13 4 10 24`	`Values of stolen items: 7 10 13 4 10 24`
	`Pirates' claim = 25`
	`Final pay back = 27`

Input:	Output (generated by our program):
`100    7 10 13 4 10 24`	`Values of stolen items: 7 10 13 4 10 24`
	`Pirates' claim = 100`
	`I'm dead!`

Practice 1 Fill the Rowboats!

Captain Jack decides to to take over a rival's ship. He needs to send his henchmen over on rowboats that can hold 6 pirates each. You will help him count out pirates in groups of 6. The last rowboat may have fewer than 6 pirates. To make your task easier each pirate has been assigned a number from 1 to N.

Input/Output Format:

The input will be N, the number of pirates you need to send over on rowboats. The output will be the number of each pirate separated by spaces, with the word "Go!" after every 6th pirate, and after the last pirate.

Example:

Sample Input 1:
10
Sample Output 1:
1 2 3 4 5 6 Go! 7 8 9 10 Go!

Sample Input 2:
18
Sample Output 2:
1 2 3 4 5 6 Go! 7 8 9 10 11 12 Go! 13 14 15 16 17 18 Go!

Practice 2 Swann's Most Wanted

After many pirate incursions, Governor Weatherby Swann wants a list of known pirates to capture. Your job is to create the list of pirates in alphabetical order, sorted by last names. If two pirates have the same last name, break ties by sorting by first names.

Input/Output Format:

The input will be N, the number of pirates, followed by additional lines with the first and last names of each pirate on each line (first names first). The output will be the sorted list of pirate names, each on a separate line.

Example:

Sample Input 1:

```
3
Black Bellamy
Edward Teach
Stede Bonnet
```

Sample Output 1:

```
Black Bellamy
Stede Bonnet
Edward Teach
```

Sample Input 2:

```
5
John Bowen
Thomas Booth
Black Bart
George Booth
Nicholas Brown
```

Sample Output 2:

```
Black Bart
George Booth
Thomas Booth
John Bowen
Nicholas Brown
```

Practice 3 Shipboard Wedding

Will Turner and Elizabeth Swann are thinking of possible seating arrangements for a shipboard wedding. Given a list of guests, they would like to come up with all possible table arrangements of 3 guests (tables are small on a ship). The order of guests at a table is not important, just which guests are selected.

Once all possible table arrangements are found, the guests in each arrangement are sorted in alphabetical order, and the arrangements are then sorted in lexicographical order.

You can compare two lists X and Y lexicographically by comparing the elements of the lists in order, using later elements to break ties if needed. For instance, a triple (a_1, a_2, a_3) is lexicographically smaller than (b_1, b_2, b_3) if

- $a_1 < b_1$, or
- $a_1 = b_1$ and $a_2 < b_2$, or
- $a_1 = b_1, a_2 = b_2$ and $a_3 < b_3$.

Input/Output Format:

The input will be the number of guests, followed by the name of each guest (in any order) on the next line. The output should be the different possible table arrangements in lexicographical order, with each arrangement on a separate line (names separated by spaces).

Example:

Sample Input 1:

```
4
Jack Gibbs Cotton Norrington
```

Sample Output 1:

```
Cotton Gibbs Jack
Cotton Gibbs Norrington
Cotton Jack Norrington
Gibbs Jack Norrington
```

Sample Input 2:

```
5
Ragetti Weatherby Davy Pintel Marty
```

Sample Output 2:

```
Davy Marty Pintel
Davy Marty Ragetti
Davy Marty Weatherby
Davy Pintel Ragetti
Davy Pintel Weatherby
Davy Ragetti Weatherby
Marty Pintel Ragetti
Marty Pintel Weatherby
Marty Ragetti Weatherby
Pintel Ragetti Weatherby
```