Question - 1
Counterfeit Currency

SCORE: **75 points**

Implementation | Strings | Medium

A counterfeit currency printer is operating in country XYZ, and all of the banks must try to identify the bad currency. Every note has a serial number that can be used to determine whether it is valid. The serial number also can be used to determine the denomination of the note. A valid serial number will have the following characteristics:

1. There are 10 to 12 characters
2. The first 3 characters are distinct uppercase English letters
3. The next 4 characters represent the year the note was printed and will always be between 1900 and 2019 inclusive.
4. The next characters represent the currency denomination and may be any one of {10, 20, 50, 100, 200, 500, 1000}
5. The last character is an uppercase English letter

Given an array of serial numbers, determine the value of the valid currency.

For example, serial numbers for *n = 8* notes are shown below.

```
Serial No.      Test 1 Test 2 Test 3 Test 4  Test
5 Valid Amt
AVG190420T        ✔      ✔      ✔      ✔       ✔
20
RTF200010002      ✔      ✔      ✔      ✔       ✔
1000
QWER201850G       ✔      ✔      ✗      ✔
✔               there is an R where the year is
supposed to start
AFA199620E        ✔      ✗      ✔      ✔       ✔
        first three characters are not distinct
ERT1947200T       ✔      ✔      ✔      ✔       ✔
200
RTY20202004       ✔      ✔      ✗      ✔       ✗
the year 2020 is out of bounds, the last character
is not an uppercase letter
DRV1984500Y       ✔      ✔      ✔      ✔       ✔
500
ETB2010400G       ✔      ✔      ✗      ✔       ✗
there are no bills for 400 denomination
```

In total, there are valid bills worth 20 + 1000 + 200 + 500 = 1720.

Function Description

Complete the function *countCounterfeit* in the editor below. The function must return an integer sum of values of valid currency.

countCounterfeit has the following parameter(s):

serialNumber[serialNumber[0],...serialNumber[n-1]]: an array of strings

Help

Constraints

- $0 < n \leq 10^5$
- $1 \leq |serialNumber[i]| \leq 14$

▼ Input Format For Custom Testing

The first line contains an integer, *n*, the number of elements in *serialNumber*.

Each line *i* of the *n* subsequent lines (where $0 \leq i < n$) contains a string, *serialNumber[i]*.

▼ Sample Case 0

Sample Input For Custom Testing

```
5
A201550B
ABB19991000Z
XYZ200019Z
ERF200220
SCD203010T
```

Sample Output

```
0
```

Explanation

Here, all the serial numbers are invalid. The total value is 0.

A201550B is invalid because the serial number does not start with three distinct uppercase English letters and the serial number is too short.

ABB19991000Z is invalid because the first three characters are not distinct.

XYZ200019Z is invalid because 19 is not a valid denomination of a note.

ERF200220 is invalid because the serial number does not end with an uppercase English letter and is too short.

SCS20180T is invalid because 2030 is not a valid year and the serial number is too short.

▼ Sample Case 1

Sample Input For Custom Testing

```
6
QDB2012R20B
RED190250E
RFV201111T
TYU20121000E
AAA198710B
AbC200010E
```

Sample Output

```
1050
```

Explanation

QDB2012R20B is invalid because there is an extra letter between the year and denomination substrings.

RED190250E is valid and its value is 50.

RFV201111T is invalid because 11 is not a valid denomination of a note.

TYU20121000E is valid and its value is 1000.

AAA198710B is invalid because the first three characters are not distinct.

AbC200010E is invalid because the second character is not uppercase.

Question - 2

Gaps in Traffic

SCORE: **50 points**

Problem Solving | Data Structures | Arrays | Pointers | Easy

Given a 2 lane road with *n* positions and a total number of *m* cars moving from left to right, from a start position to a finish position, determine the largest gap in positions of all cars, without regard to lanes.

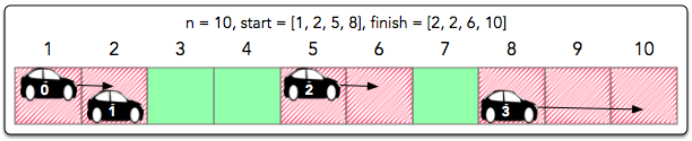
Example

n = 10

start = [1, 2, 5, 8]

finish = [2, 2, 6, 10]

The following is a graphical representation of a snapshot of the cars on the road. *start* and *finish* mark the positions of the rear and front of each car. The length of the road is *n*.



Positions that contain cars are *red*, and positions that are open are solid *green*.

~~In the diagram above, there are two gaps. One has length *2* and spans from marker *3* to *4*. The other has length *1* and spans marker *7*. In this scenario, the widest gap is *2*.~~

The diagram above depicts a road *n = 10* units in length with *m = 4* cars :

- The first car spans from position start[0] = 1 to position finish[0] = 2
- The second car spans from position start[1] = 2 to position finish[1] = 2
- The third car spans from position start[2] = 5 to position finish[2] = 6
- The fourth car spans from position start[3] = 8 to position finish[3] = 10
- There are gaps at positions 3-4 and 7. The largest gap between cars is 2.

Function Description

Complete the function *widestGap* in the editor below.

widestGap has the following parameter(s):

int n: integer, the length of the road section

int start[m]: integers that represent the positions of the rear of each car

int finish[m]: integers that represent the positions of the front of each car

Returns:

int: an integer that denotes the length of the longest gap between cars.

Constraints

- $1 \le n \le 10^9$
- $1 \le m \le 10^5$
- $1 \le start[i] \le finish[i] \le n$, where $0 \le i < m$.

▼ Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer *n*, the length of the road*s*.
The second line contains an integer *m*, the number of cars.
The next *m* lines each contain an element *start[i]*.
The next line contains an integer *m*, the number of cars.
The next *m* lines each contain an element *finish[i]*.

▼ Sample Case 0

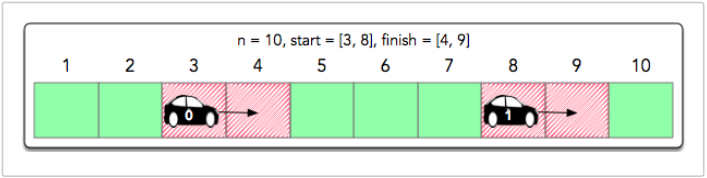
Sample Input

```
STDIN    Function
-----    --------------------
10    →  n = 10
2     →  start[] size m = 2
3     →  start = [3, 8]
8
2     →  finish[] size m = 2
4     →  finish = [4, 9]
9
```

Sample Output 0

```
3
```

Explanation 0



The diagram above depicts a road *n = 10* units in length with *m = 2* cars :

- The first car spans from position start[0] = 3 to position finish[0] = 4
- The second car spans from position start[1] = 8 to position finish[1] = 9
- No car is in positions [1,2], positions [5, 6, 7] or position [10]. The largest gap between cars is 3.

▼ **Sample Case 1**

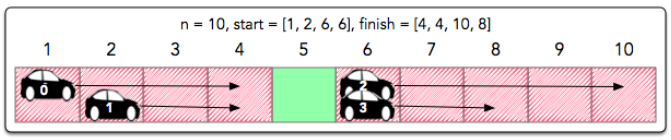
Sample Input1

```
STDIN    Function Parameters
-----    -------------------
10    →  n = 10
4     →  start[] size m = 4
1     →  start = [1, 2, 6, 6]
2
6
6
4     →  finish[] size m = 4
4     →  finish = [4, 4, 10, 8]
4
10
8
```

Sample Output 1

```
1
```

Explanation 1



The diagram above depicts a road *n = 10* units in length with *m = 4* cars :

- The first car spans from position start[0] = 1 to position finish[0] = 4
- The second car spans from position start[1] = 2 to position finish[1] = 4
- The third car spans from position start[2] = 6 to position finish[2] = 10
- The fourth car spans from position start[3] =6 to position finish[3] = 8
- No car is in position 5. The largest gap between cars is 1.

Question - 3

SCORE: **50 points**

Table of Contents

Easy | Strings | Interviewer Guidelines

Create a table of contents for a simple markup language. It must follow two rules:

1. If a line starts with a single # followed by a space, then it's a chapter title.
2. If a line starts with a double # followed by a space, then it's a section title.

The table of contents should be displayed in the following format:

```
1. Title of the first chapter
1.1. Title of the first section of the first
chapter
1.2. Title of the second section of the first
chapter
...
2. Title of the second chapter
2.1. Title of the first section of the second
chapter
2.2. Title of the second section of the
second chapter
...
```

Note that each number is followed by a period and the last period is followed by 1 space.

For example, the input text is *n = 12* lines long, where *text* is the following:

```
# Algorithms
This chapter covers the most basic algorithms.
## Sorting
Quicksort is fast and widely used in practice
Merge sort is a deterministic algorithm
## Searching
DFS and BFS are widely used graph searching
algorithms
Some variants of DFS are also used in game theory
applications
# Data Structures
This chapter is all about data structures
It's a draft for now and will contain more
sections in the future
# Binary Search Trees
```

This is the table of contents that must be produced:

```
1. Algorithms
1.1. Sorting
1.2. Searching
2. Data Structures
3. Binary Search Trees
```

Function Description

Complete the function *tableOfContents* in the editor below.

tableOfContents has the following parameter:

string *text[n]*: the input text

Returns

string[]: each string is a line in the table of contents

Constraints

- $1 \le n \le 1000$
- 1 ≤ length of *text[i]* ≤ 100
- When a line starts with # or with ##, these special characters are always followed by a space.
- The first line of the text is guaranteed to be a chapter line.

▼ Input Format Format for Custom Testing

In the first line, there is a single integer, *n*, the number of lines in *text*.

Each line *i* of the *n* subsequent lines (where $0 \le i < n$) contains a string, *text[i]*.

▼ Sample Case 0

Sample Input

```
10
# Cars
Cars came into global use during the 20th century
Most definitions of car say they run primarily on
roads
## Sedan
Sedan's first recorded use as a name for a car
body was in 1912
## Coupe
A coupe is a passenger car with a sloping rear
roofline and generally two doors
## SUV
The predecessors to SUVs date back to military
and low-volume models from the late 1930s
There is no commonly agreed definition of an SUV,
and usage varies between countries.
```

Sample Output

```
1. Cars
1.1. Sedan
1.2. Coupe
1.3. SUV
```

Explanation

The first line of input indicates there are *n = 10* lines of text. There is only 1 chapter in the input, and it contains 3 sections. All the lines that don't begin with # or ## are ignored in the table of contents.

▼ Sample Case 1

Sample Input

```
10
# Games
## Board
## Computer
## Zero sum
## Multiplayer
# Strategies
## Greedy
## Tree pruning
## Others
# Summary
```

Sample Output

```
1. Games
1.1. Board
1.2. Computer
1.3. Zero sum
1.4. Multiplayer
2. Strategies
2.1. Greedy
2.2. Tree pruning
2.3. Others
3. Summary
```

Explanation

Again, the first line of input indicates there are *n = 10* lines of text. This text already looks like an outline because it contains only

chapters and sections. Chapter 1 has 4 sections in it, Chapter 2 has 3 sections, and Chapter 3 has no sections.

Question - 4

SCORE: **75 points**

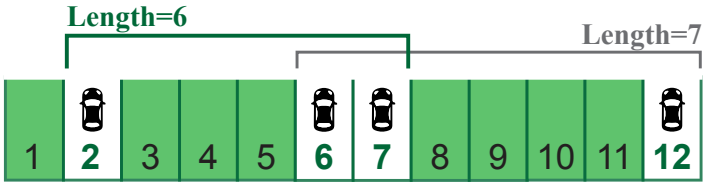
Parking Dilemma

Medium | Binary Search | Algorithms | Theme: Automotive

There are many cars parked in a parking lot. The parking lot is a straight line with a parking spot for every meter. There are *n* cars currently parked and a roofer wants to cover them with a roof. The requirement is that at least *k* cars are covered by the roof. Determine the minimum length of the roof that will cover *k* cars.

Example

n = 4
cars = [6, 2, 12, 7]
k = 3



Two roofs that cover three cars are possible: one covering spots *2* through *7* with a length of *6*, and another covering slots *6* through *12* with a length of *7*. The shortest roof that meets the requirement is of length *6*.

Function Description

Complete the function *carParkingRoof* in the editor below.

carParkingRoof has the following parameter(s):
int cars[n]: the parking spots where cars are parked
int k:the number of cars that have to be covered by the roof

Returns:
int: the minimum length of a roof that can cover *k* cars

Constraints

- $1 \le n \le 10^5$
- $1 \le k \le n$
- $1 \le cars[i] \le 10^{14}$
- All spots taken by cars are unique

▼ Input Format Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

In the first line, there is a single integer, n, the size of *cars[]*.
Then, *n* line follows. In the i^{th} of them, there is a single integer *cars[i]*.
In the last line, there is a single integer *k*.

▼ Sample Case 0

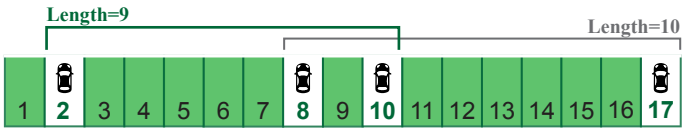
Sample Input

```
STDIN       Function
-----       --------
4        →  cars[] size n = 4
2        →  cars[] = [ 2, 10, 8, 17 ]
10
8
17
3        →  k = 3
```

Sample Output

```
9
```

Explanation



A roof can be built of length *9* covering all parking spots from the *2nd* through the *10th*, so covering *3* cars in spots *2*, *8*, and *10*. There is no shorter roof that can cover *3* cars.

▼ Sample Case 1

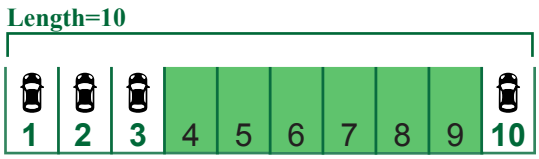
Sample Input

```
STDIN     Function
-----     --------
4       → cars[] size n = 4
1       → cars[] = [ 1, 2, 3, 10 ]
2
3
10
4       → k = 4
```

Sample Output

```
10
```

Explanation



All of the cars must be covered. The shortest roof that can cover them has a length of *10* and starts at slot *1* and ends at slot *10*.

Question - 5

Freelancing Platform

SCORE: **50 points**

Easy | Implementation | Algorithms | Interviewer Guidelines

A customer has posted several web development projects on a freelancing platform, and various web developers have put in bids for these projects. Given the bid amounts and their corresponding projects, what is the minimum amount the customer can pay to have all the projects completed?

Note: If any project has no bids, return -1.

Example
numProjects = 3 projects.
projectId = [2, 0, 1, 2]
bid = [8, 7, 6, 9].

- *projectId[i]* is aligned with *bid[i]*
- The first web developer bid 8 currency units for project 2.
- The second web developer bid 7 currency units for project 0.
- The third web developer bid 6 currency units for project 1.
- The fourth web developer bid 9 currency units for project 2.

There is only one choice of who to hire for project *0*, and it will cost *7*. Likewise, there is only one choice for project *1*, which will cost *6*. For project *2*, it is optimal to hire the first web developer, instead of the fourth, and doing so will cost *8*. So the final answer is *7 + 6 + 8 = 21*.

If instead there were *n = 4* projects, the answer would be *-1* since there were no bids received on the fourth project.

Function Description
Complete the function *minCost* in the editor below.

minCost has the following parameters:
int numProjects: the total number of projects posted by the client (labeled from *0* to *n*)
int projectId[n]: an array of integers denoting the projects that the freelancers bid on
int bid[n]: an array of integers denoting the bid amounts posted by the freelancers

Returns:
long: the minimum cost the client can spend to complete all projects, or *-1* if any project has no bids.

Constraints
- $1 \le numProjects, n \le 5 \times 10^5$
- $0 \le projectId[i] < n$
- $1 \le bid[i] \le 10^9$

▼ Input Format For Custom Testing

The first line contains an integer, *numProjects*.
The next line contains an integer, *n*, which denotes the number of elements in array *projectId[]*.
Each line *i* of the *n* subsequent lines (where $0 \le i < n$) contains an integer, *projectId[i]*.
The next line contains an integer, *n*, which denotes the number of elements in array *bid*.

Each line *i* of the *n* subsequent lines (where *0 ≤ i < n*) contains an integer, *bid[i]*.

▼ Sample Case 0

Sample Input

```
STDIN       Function
-----       -----
2     →     numProjects = 2
5     →     projectId[] size n = 5
0     →     projectId = [0, 1, 0, 1, 1]
1
0
1
1
5     →     bid[] size n = 5
4     →     bid = [4, 74, 47, 744, 7]
74
47
744
7
```

Sample Output

```
11
```

Explanation

The bids are as follows:

- The first web developer bid *4* currency units for project *0*.
- The second web developer bid *74* currency units for project *1*.
- The third web developer bid *47* currency units for project *0*.
- The fourth web developer bid *744* currency units for project *1*.
- The fifth web developer bid *7* currency units for project *1*.

The optimal solution is to hire the first web developer to complete project *0* (which costs *4*) and to hire the fifth web developer to complete project *1* (which costs *7*). This brings the total cost to *11*.

▼ Sample Case 1

Sample Input

```
STDIN       Function
-----       -----
2     →     numProjects = 2
2     →     projectId[] size n = 2
1     →     projectId = [1, 1]
1
2     →     bid[] size n = 2
4     →     bid = [4, 7]
7
```

Sample Output

```
-1
```

Explanation

Because there are no bids for project *0*, the function should return *-1*.

Question - 6

SCORE: **50 points**

Programming Contest

Easy | Sorting | Interviewer Guidelines

A programming organization is planning a contest for several programmers, each of which has a certain rating. (The higher the rating, the better the programmer.) Each programmer is paired with another programmer, and the difference between their ratings is referred to as the "bias amount". Given the ratings of all the programmers in the contest, what is the minimum total bias amount that can be achieved by optimally planning the programmer pairs?

Example
n = 4
ratings = [4, 2, 5, 1]

The optimal solution is:
Pair the second programmer (2) with the fourth (1) for a difference of 1.
Pair the first programmer (4) with the third (5) for a difference of 1.
This results in a total bias amount of 1 + 1 = 2.

Function Description
Complete the function *minimizeBias* in the editor below.

minimizeBias has the following parameter:
int *ratings[n]:* the ratings of each of the programmers
Returns:
int: the minimum total bias amount that can be achieved in the contest

Constraints

- $1 \le n \le 10^5$
- $1 \le ratings[i] \le 10^9$
- *n* is even.

▼ Input Format For Custom Testing

The first line contains an integer, *n*, the number of elements in *ratings*.
Each line *i* of the *n* subsequent lines (where $0 \le i < n$) contains an integer, *ratings[i]*.

▼ Sample Case 0

Sample Input For Custom Testing

```
STDIN       Function
-----       --------
4       →   ratings[] size n = 4
1       →   ratings = [1, 3, 6, 6]
3
6
6
```

Sample Output

```
2
```

Explanation

The optimal solution is to pair the first programmer (1) with the second (3) for a difference of 2, and the third programmer (6) with the fourth (6) for a difference of 0. This results in a total bias amount of 2.

▼ Sample Case 1

Sample Input For Custom Testing

```
STDIN    Function
-----    --------
6    →   ratings[] size n = 6
2    →   ratings = [2, 4, 5, 3, 7, 8]
4
5
3
7
8
```

Sample Output

```
3
```

Explanation

The optimal solution is to assign the following pairs: (2,3), (4,5), and (7,8). This results in the least total bias amount, which is 3.

Question - 7

Football Scores

SCORE: **75 points**

Data Structures | Medium | Binary Search | Algorithms | Arrays | Problem Solving

The number of goals achieved by two football teams in matches in a league is given in the form of two lists. For each match of team B, compute the total number of matches of team A where team A has scored *less than or equal to* the number of goals scored by team B in that match.

Example

teamA = [1, 2, 3]

teamB = [2, 4]

Team A has played three matches and has scored *teamA = [1, 2, 3]* goals in each match respectively. Team B has played two matches and has scored *teamB = [2, 4]* goals in each match respectively. For *2* goals scored by team B in its first match, team A has *2* matches with scores *1* and *2*. For *4* goals scored by team B in its second match, team A has *3* matches with scores *1*, *2* and *3*. Hence, the answer is *[2, 3]*.

Function Description

Complete the function *counts* in the editor below.

counts has the following parameter(s):

int teamA[n]: first array of positive integers

int teamB[m]: second array of positive integers

Return

int[m]: an array of *m* positive integers, one *for each teamB[i]* representing the total number of elements from *teamA[j]* satisfying *teamA[j] ≤ teamB[i]* where *0 ≤ j < n* and *0 ≤ i < m*, in the given order.

Constraints

- $2 \le n, m \le 10^5$
- $1 \le teamA[j] \le 10^9$, where $0 \le j < n$.
- $1 \le teamB[i] \le 10^9$, where $0 \le i < m$.

▼ Input Format For Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer *n*, the number of elements in *teamA*.
The next *n* lines each contain an integer describing *teamA[j]* where $0 \le j < n$.
The next line contains an integer *m*, the number of elements in *teamB*.
The next *m* lines each contain an integer describing *teamB[i]* where $0 \le i < m$.

▼ Sample Case 0

Sample Input 0

```
STDIN       Function
-----       --------
4       →   teamA[] size n = 4
1       →   teamA = [1, 4, 2, 4]
4
2
4
2       →   teamB[] size m = 2
3       →   teamB = [3, 5]
5
```

Sample Output 0

```
2
4
```

Explanation 0

Given values are $n = 4$, $teamA = [1, 4, 2, 4]$, $m = 2$, and $teamB = [3, 5]$.

1. For $teamB[0] = 3$, we have 2 elements in $teamA$ ($teamA[0] = 1$ and $teamA[2] = 2$) that are $\le teamB[0]$.
2. For $teamB[1] = 5$, we have 4 elements in $teamA$ ($teamA[0] = 1$, $teamA[1] = 4$, $teamA[2] = 2$, and $teamA[3] = 4$) that are $\le teamB[1]$.

Thus, the function returns the array $[2, 4]$ as the answer.

▼ Sample Case 1

Sample Input 1

```
STDIN     Function
-----       --------
5       →   teamA[] size n = 5
2       →   teamA = [2, 10, 5, 4, 8]
10
5
4
8
4       →   teamB[] size m = 4
3       →   teamB = [3, 1, 7, 8]
1
```

```
7
8
```

Sample Output 1

```
1
0
3
4
```

Explanation 1

Given values are *n = 5*, *teamA = [2, 10, 5, 4, 8]*, *m = 4*, and *teamB = [3, 1, 7, 8]*.

1. For *teamB[0] = 3*, we have *1* element in *teamA* (*teamA[0] = 2*) that is ≤ *teamB[0]*.
2. For *teamB[1] = 1*, there are *0* elements in *teamA* that are ≤ *teamB[1]*.
3. For *teamB[2] = 7*, we have *3* elements in *teamA* (*teamA[0] = 2*, *teamA[2] = 5*, and *teamA[3] = 4*) that are ≤ *teamB[2]*.
4. For *teamB[3] = 8*, we have *4* elements in *teamA* (*teamA[0] = 2*, *teamA[2] = 5*, *teamA[3] = 4*, and *teamA[4] = 8*) that are ≤ *teamB[3]*.

Thus, the function returns the array *[1, 0, 3, 4]* as the answer.

Question - 8

SCORE: **50 points**

Compliance Priorities

Easy | Iteration | Algorithms | Problem Solving

A system used by a compliance department contains a queue of all current compliance issues along with their priorities. The priorities range from *1* to *99*. Create an algorithm that will reassign priorities so that the value of the maximum priority assigned is minimized, keeping the relative priorities between all issues the same.

Example

priorities = [1, 4, 8, 4]

There are three priority levels: 1, 4 and 8. The array elements are reassigned to priorities *[1, 2, 3, 2]*. Their relative priorities are maintained while the value of the maximum priority is minimized.

Given the priorities of the issues, return a list that contains the reassigned priority values without reordering.

Function Description

Complete the *reassignedPriorities* function in the editor below. It must return an integer array that represents the reassigned priorities of each element in the original order.

reassignedPriorities has the following parameter(s):

int priorities[n]: an array of integers that represents current priorities

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq priorities[i] \leq 99$

▼ Input Format For Custom Testing

The first line contains an integer, *n*, that denotes the size of *priorities*.
Each line *i* of the *n* subsequent lines (where $0 \leq i < n$) contains *priorities[i]*.

▼ Sample Case 0

Sample Input For Custom Testing

```
STDIN    Function
-----    --------
4    →   priorities[] size n = 4
1    →   priorities = [1, 3, 7, 3]
3
7
3
```

Sample Output

```
1
2
3
2
```

Explanation

There are three priority levels: 1, 3, 7. They can be reassigned priorities of 1, 2 and 3 respectively. Replacing the values in *priorities* returns *[1, 2, 3, 2]*.

▼ Sample Case 1

Sample Input For Custom Testing

```
STDIN    Function
-----    --------
5    →   priorities[] size n = 5
2    →   priorities = [2, 9, 3, 2, 3]
9
3
2
3
```

Sample Output

```
1
3
2
1
2
```

Explanation

There are three priority levels: 2, 3, 9. These can be reassigned priorities of 1, 2 and 3 respectively. Replacing the values in *priorities* returns *[1, 3, 2, 1, 3]*.

Question - 9

SCORE: **75 points**

Profit Targets

Binary Search | Data Structures | Medium | Algorithms | Arrays | Problem Solving | Theme: Finance

A financial analyst is responsible for a portfolio of profitable stocks represented in an array. Each item in the array represents the yearly profit of a corresponding stock. The analyst gathers all distinct pairs of stocks that reached the target profit. Distinct pairs are pairs that differ in at least one element. Given the array of profits, find the number of distinct pairs of stocks where the sum of each pair's profits is exactly equal to the target profit.

Example

stocksProfit = [5, 7, 9, 13, 11, 6, 6, 3, 3]

target = 12 profit's target

- There are *4* pairs of stocks that have the sum of their profits equals to the target 12 . Note that because there are two instances of *3* in *stocksProfit* there are two pairs matching (9, 3): *stocksProfits* indices 2 and 7, and indices 2 and 8, but only one can be included.
- There are *3* distinct pairs of stocks: *(5, 7)*, *(3, 9)*, and *(6, 6)* and the return value is *3*.

Function Description

Complete the function *stockPairs* in the editor below.

stockPairs has the following parameter(s):

int stocksProfit[n]: an array of integers representing the stocks profits

target: an integer representing the yearly target profit

Returns:

int: the total number of pairs determined

Constraints

- $1 \le n \le 5 \times 10^5$
- $0 \le stocksProfit[i] \le 10^9$
- $0 \le target \le 5 \times 10^9$

▼ Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer *n*, the size of the array *stocksProfit*.

The next *n* lines each contain an element *stocksProfit[i]* where $0 \le i < n$.

The next line contains an integer *target*, the target value.

▼ Sample Case 0

Sample Input 0

```
STDIN       Function
-----       --------
6       →   stocksProfit[] size n = 6
1       →   stocksProfit = [1, 3, 46, 1, 3, 9]
3
46
1
3
9
47      →   target = 47
```

Sample Output 0

```
1
```

Explanation 0

There are *4* pairs where *stocksProfit[i] + stocksProfit[j] =* 47

1. *(stocksProfit0] = 1, stocksProfit[2] = 46)*
2. *(stocksProfit[2] = 46, stocksProfit[0] = 1)*
3. *(stocksProfit[2] = 46, stocksProfit[3] = 1)*
4. *(stocksProfit[3] = 1, stocksProfit[2] = 46)*

Since all four pairs contain the same values, there is only *1 distinct* pair of stocks : *(1, 46)*.

▼ Sample Case 1

Sample Input 1

```
STDIN       Function
-----       --------
7       →   stocksProfit[] size n = 7
6       →   stocksProfit = [6, 6, 3, 9, 3, 5, 1]
6
3
9
3
5
1
12      →   target = 12
```

Sample Output 1

```
2
```

Explanation 1

There are *5* pairs where *stocksProfit[i] + stocksProfit[j] =* 12:

1. *(stocksProfit[0] = 6, stocksProfit[1] = 6)*
2. *(stocksProfit[1] = 6, stocksProfit[0] = 6)*
3. *(stocksProfit[2] = 3, stocksProfit[3] = 9)*
4. *(stocksProfit[3] = 9, stocksProfit[2] = 3)*
5. *(stocksProfit[3] = 9, stocksProfit[4] = 3)*
6. *(stocksProfit[4] = 3, stocksProfit[3] = 9)*

The first *2* pairs are the same, as are the last *4*. There are only *2 distinct* pairs of stocks: *(3, 9)* and *(6, 6)*.

Question - 10

Diverse Deputations

SCORE: **50 points**

Combinatorics | Easy | Math | Problem Solving | Interviewer Guidelines

A professional society is using a program to determine possible diverse deputations of 3 members for an upcoming conference. There are *m* men and *w* women who are eligible. A deputation is diverse only if it contains at least one man and at least one woman. Two deputations are considered distinct if one has a member that the other does not. Given a number of men and women, determine the number of distinct ways to select a diverse deputation of *3* people.

Example

m = 1

w = 3

There is *m = 1* man available and there are *w = 3* women. Label them *m1*, *w1*, *w2*, *w3* for demonstration. There are *3* possible ways to form a diverse deputation: (*m1, w1, w2)*, (*m1, w1, w3*) and (*m1, w2, w3*). The only other possible permutation is (*w1, w2, w3*), which does not include a man, so it is invalid.

Function Description

Complete the function *diverseDeputation* in the editor below.

diverseDeputation has the following parameters:

int m: the number of men available

int w: the number of women available

Returns

int: the number of ways to select a diverse deputation from *m* men and *w* women

Constraints

- *0 ≤ m, w ≤ 1000*

▼ Input Format For Custom Testing

The first line contains an integer *m*, that denotes the number of men.

The second line contains an integer *w*, that denotes the number of women.

▼ Sample Case 0

Sample Input For Custom Testing

```
STDIN    Function
-----    --------
3     →  number of men m = 3
0     →  number of women w = 0
```

Sample Output 0

```
0
```

Explanation 0

The number of women is *0* so there is no way to select a diverse deputation.

▼ Sample Case 1

Sample Input For Custom Testing

```
STDIN    Function
-----    --------
2     →  number of men m = 2
2     →  number of women w = 2
```

Sample Output 1

```
4
```

Explanation 1

In this case, *m = 2* and *w = 2*. This yields *4* ways to select a diverse deputation: (*m1, w1, w2*), (*m1, m2, w2*), (*m2, w1, w2*), (*m1, m2, w1*).