**Bài 1:**

**Level: Dễ**

**Yêu cầu:**

Cho một mảng số nguyên Hãy kiểm tra xem tần số xuất hiện của tất cả các số nguyên trong mảng có bằng nhau hay không

**Input:** 1 chuỗi chứa dãy số nguyên, mỗi số cách nhau bởi 1 dấu “,”

**Output:** 1 giá trị Boolean

**Ví dụ:**

* Với inputArray = [1, 2, 2, 1], thì kết quả checkEqualFrequency(inputArray) = true  
  Số 1 và số 2 đều có tần số xuất hiện là 2
* Với inputArray = [1, 2, 2, 3, 1, 3, 1, 3], thì kết quả checkEqualFrequency(inputArray) = false.  
  Số 1 có tần số xuất hiện là 3, khác vs số 2 và số 3 có tần số xuất hiện là 2 .

**Test case:**

* **Input**: 1, 2, 2, 1   
  **Output**: true
* **Input**: 1, 2, 2, 3, 1, 3, 1, 3

**Output**: false

* **Input**: 4, 3, 2, 1, 4, 2, 3, 1, 1

**Output**: false

* **Input**: 4, 3, 2, 1, 4, 2, 3, 1,

**Output**: true

* **Input**: 100

**Output**: false

**Bài 2:**

**Level: Dễ**

**Yêu cầu:**

Cho hai xâu kí tự, tìm số lượng ký tự chung giữa chúng.

**Input:**

Dòng đầu tiên là 1 chuỗi ký tự

Dòng thứ 2 là 1 chuỗi ký tự khác cần so sánh với chuỗi trên

**Output:** 1 giá trị số nguyên

**Ví dụ:**

* Với s1 = "aabcc" và s2 = "adcaa", thì kết quả commonCharacterCount(s1, s2) = 3.  
  Vì xâu s1 và s2 có 3 kí tự chung: 2 kí tự 'a' và 1 kí tự 'c'.

**Test case:**

* **Input**:

aabcc

adcaa

**Ouput**: 3

* **Input**

Abcdef

Adghjklbc

**Output**: 4

* **Input**

Abcdef1234

Uykhldg1

**Output**: 2

* **Input**

A

BCd

**Output**: 0

* **Input**

Abc

ade

**Output**: 0 ( A lớn khác a)

**Bài 3:**

**Level: Dễ**

**Mô tả**

Kiểm tra mảng có “đẹp” hay không

Một mảng được gọi là "đẹp" nếu mảng đó chứa một phần tử mà phần tử đó chia mảng làm 2 phần (không rỗng và không chứa phần tử đó) có tổng bằng nhau.

**Yêu cầu**

Hãy kiểm tra xem một mảng bất kì có phải là một mảng "đẹp" hay không?

**Input:** 1 chuỗi chứa dãy số nguyên, mỗi số cách nhau bởi 1 dấu “,”

**Output**: 1 giá trị Boolean

Ví dụ:

* Với a = [1, 2, 3] thì kết quả beautifulArray(a) = false
* Với a = [1, 2, 3, 3] đầu ra của chương trình sẽ là beautifulArray(a) = true  
  Chọn phần tử ở vị trí thứ 3 (tính từ trái sang), thì mảng ban đầu được chia thành 2 phần [1,2] và [3] có tổng 2 phần bằng nhau

**Test case:**

* **Input:** 1, 2, 3

**Output:** false

* **Input:** 1, 2, 3, 3

**Output:** true

* **Input:** 1, 2, 3, 4

**Output:** false

* **Input:** 3, 2, 4, 5, 9

**Output:** true

* **Input:** 1, 2, -1, 2

**Output:** true

**Bài 4:**

**Level: Trung bình**

**Mô tả**

Chương trình sửa lại xâu kí tự:

Cho một xâu kí tự s gồm nhiều từ. Trong đó mỗi từ sẽ được bắt đầu bởi một kí tự in hoa.

Vì một lí do gì đó, người ta quên chèn các khoảng trắng giữa các từ.

**Yêu cầu**

Hãy viết chương trình sửa lại xâu kí tự đó theo yêu cầu sau:

Tách các từ, và thêm 1 kí tự cách (space) giữa 2 từ liên tiếp

Chuyển đổi các kí tự in hoa thành kí tự in thường

**Input**: 1 chuỗi ký tự

**Output**: 1 chuỗi ký tự

**Ví dụ:**

* Với s = "CodegymDaNang", thì kết quả amendTheSentence(s) = "codegym da nang"
* Với s = "Hello", thì kết quả amendTheSentence(s) = "hello"

**Test case:**

* **Input**: Hello

**Output**: hello

* **Input**: Hello CodeGym

**Output**: hello code gym

* **Input**: DaNang

**Output**: da nang

* **Input**: CodeGymDaNang

**Output**: code gym da nang

* **Input**: CodeMarathon

**Output**: code marathon

**Bài 5:**

**Level: Trung bình**

**Mô tả:**

Xếp hàng theo chiều cao tăng dần:

Một nhóm người đứng thành hàng trong công viên. Giữa họ có một số cây không thể di chuyển

Giá trị “-1” đại diện cho cây trong công viên.

**Yêu cầu:**

Nhiệm vụ của bạn là thay đổi vị trí của họ, sao cho chiều cao của họ tạo thành một dãy tăng dần (không tính cây). Chú ý rằng cây không thể di chuyển.

**Input:** 1 chuỗi chứa dãy số nguyên, mỗi số cách nhau bởi 1 dấu “,”

**Output:** 1 mảng số nguyên

**Ví dụ:**

* Với a = [-1, 150, 190, 170, -1, -1, 160, 180] thì kết quả sortByHeight(a) = [-1, 150, 160, 170, -1, -1, 180, 190].

**Test case:**

* **Input**: -1, 150, 190, 170, -1, -1, 160, 180

**Output**: [-1, 150, 160, 170, -1, -1, 180, 190].

* **Input**: -1, 150, 165, 170, -1, -1, 160, 180

**Output**: [-1, 150, 160, 165, -1, -1, 170, 180].

* **Input**: 150, -1, 190, 170, -1, -1, 160, 180

**Output**: [150, -1, 160, 170, -1, -1, 180, 190].

* **Input**: -1, 150, 190, 170, -1, -1, 160, -1

**Output**: [-1, 150, 160, 170, -1, -1, 190, -1].

* **Input**: -1, 150, -1, 170, -1, -1, 160, 180

**Output**: [-1, 150, -1, 160, -1, -1, 170, 180].

**Bài 6:**

**Level: Trung bình**

**Yêu cầu:**

Viết hàm truyền vào một chuỗi findChar(string)

Hàm trả về 1 mảng chứa các ký tự xuất hiện nhiều hơn 1 lần trong chuỗi. Nếu không có ký tự nào xuất hiện nhiều hơn 1 lần thì trả về mảng rỗng

**Input**: 1 chuỗi ký tự

**Output**: 1 mảng ký tự

Ví dụ:

* findChar (“abcbdcef”) => [b,c]
* findChar (“abdcef”) => []

**Test case:**

* **Input**: abcbdcef

**Output**: [b,c]

* **Input**: abdcef

**Output**: []

* **Input**: abdcef123d

**Output**: [d]

* **Input**: abdceffecdba

**Output**: [a,b,c,d,e,f]

* **Input**: a

**Output**: []

**Bài 7:**

**Level: Dễ**

**Yêu cầu:**

Viết hàm truyền vào một mảng 2 chiều (n x n) chứa các số nguyên bất kỳ

Hàm sẽ trả về 1 mảng (1 chiều )các số nguyên tố đã được sắp xếp tăng dần. Với các phần tử là các số nguyên tố xuất hiện trong mảng 2 chiều.

**Input**: Gồm n dòng. Với mỗi dòng là 1 chuỗi các số nguyên, mỗi số cách nhau bởi dấu “,”

**Output** : 1 mảng 1 chiều số nguyên tố đã được sắp xếp

**Ví dụ:**

* Với input

[

  [1,2,8,43],

 [9,4,11,7],

 [11,6,4,0],

 [1,2,6,51]

] thì output nhận được là: [2,7,11,43]

**Test case:**

* **Input**

[

  [1,2,8,43],

 [9,4,11,7],

 [11,6,4,0],

 [1,2,6,51]

]

**Output**: [2,7,11,43]

* **Input**

[

  [1,2,8,43],

 [9,49,11,7],

 [11,23,4,0],

 [1,2,6,51]

]

**Output**: [2 ,7 ,11 ,23 ,43]

* **Input**

[

  [1,2,8,43],

 [9,4,11,8],

 [11,6,4,0],

 [1,2,6,51]

]

**Output**: [2,11,43]

* **Input**

[

  [1,10,8,44],

 [9,4,12,8],

 [16,6,4,0],

 [1,4,6,51]

]

**Output**: []

* **Input**

[

  [1,3,8,43],

 [9,4,11,7],

 [11,6,4,0],

 [1,2,6,51]

]

**Output**: [2,3,7,11,43]

**Bài 8:**

**Level: Trung bình**

**Yêu cầu:**

Tìm số lần xuất hiện mảng con trong mảng cha

Điều kiện child là con của parent (các phần tử trong mảng child xuất hiện ít nhất 1 lần trong mảng parent)

**Input**:

Dòng đầu tiên là 1 chuỗi các số nguyên của mảng parent, mỗi số nguyên sẽ cách nhau bởi dấu “,”

Dòng thứ 2 là 1 chuỗi các số nguyên của mảng child, mỗi số nguyên sẽ cách nhau bởi dấu “,”

**Output**: Số lần xuất hiện của mảng con trong mảng cha

**Ví dụ:**

* parent =[1,4,5,2,1],child=[1,2,3] =>return 0 // không xuất hiện
* parent =[1,4,5,2,3] child=[1,2,3] => return 1; // xuất hiện 1 lần

**Test case:**

* **Input**:

1,4,5,2,1

1,2,3

**Output**: 0

* **Input**:

1,4,5,2,3

1,2,3

**Output**: 1

* **Input**:

1,4,5,2,3,2,3,1

1,2,3

**Output**: 2

* **Input**:

1,4,5,2,3,2,3,8

1,2,3

**Output**: 1

* Input:

1,2,3

1,1,1

Output: 1

**Bài 9:**

**Level: Dễ**

**Mô tả:**

Người ta định nghĩa số vòng là một số tự nhiên mà mà chữ số đầu trùng với chữ số cuối của nó. Ví dụ 303, 101, 8, 88 là những số vòng.

**Yêu cầu:**

Cho 2 số nguyên dương L, R, hãy đếm số lượng số vòng trong đoạn [L; R]

**Input**:

Dòng đầu tiên là 1 số nguyên L

Dòng thứ 2 là 1 số nguyên R

**Output**: Số lượng của số vòng trong khoảng [L;R]

**Ví dụ:**

* Với L = 9, R = 13 thì circlenum(L,R) = 2.  Trong đoạn [9; 13] có 2 số vòng là 9 và 11

**Test case:**

* **Input**:

9

13

**Output**: 2.

* **Input**:

10

100

**Output**: 9.

* **Input**:

10

30

**Output**: 2.

* **Input**:

5

25

**Output**: 7.

* **Input**:

10

20

**Output**: 1.

**Bài 10:**

**Level: Trung bình**

**Mô tả:**

Bình đang có n VNĐ và bằng cách nào đó Bình biết trước được giá mua và giá bán của 1 đô-la của các ngày tiếp theo. Biết giá mua và giá bán đô-la tại 1 ngày là như nhau. Bình muốn tìm cách mua và bán đô-la hợp lý nhất để vào ngày cuối cùng số VNĐ Bình có là lớn nhất **biết Bình chỉ có thể mua đô-la tại đúng 1 ngày và bán cũng vậy.**

**Yêu cầu:**

Hãy viết hàm trả về số đô-la tối đa có thể thu được vào ngày cuối cùng

**Input**:

Dòng đầu tiên là 1 chuỗi các số nguyên là giá mua và bán của các ngày tiếp theo, mỗi số nguyên sẽ cách nhau bởi dấu “,”.

Dòng thứ 2 là n với n là số tiền Bình có

**Output**: 1 giá trị số nguyên.

**Ví dụ**

* Cho arr = [3, 7], n = 4, output sẽ có dạng exchangeMoney(arr, n) = 8.

Giải thích:

arr = [3, 7] có nghĩa là giá mua và bán đô-la tại ngày 1 và 2 lần lượt là 3 và 7

Bình sẽ mua vào 3 đô-la tại ngày 1 và bán ra với giá 7 đô-la vào ngày 2. Tổng số VNĐ Bình thu được là: 4 - 3 + 7 = 8.

* Cho arr = [3, 2, 1], n = 10, output sẽ có dạng exchangeMoney(arr, n) = 10.  
   Giải thích: Bình sẽ không thực hiện mua và bán đô-la vì nó sẽ đem lại lỗ nên số VNĐ Bình có vẫn giữ nguyên là 10.

**Test case:**

* **Input**:

3, 7

4

**Output**: 8

* **Input**:

3, 2, 1

10

**Output**: 10

* **Input**:

3, 2, 1, 8

5

**Output**: 12

* **Input**:

3, 5, 2, 7, 1

7

**Output**: 12

* **Input**:

3, 2, 1, 0, 3

10

**Output**: 12

**Bài 11:**

**Level: Trung bình**

**Mô tả:**

Thầy Ba muốn có s cây bút để tặng cho học sinh của mình.

Ở cửa hàng bán bút CODEGYM đang có khuyến mãi như sau: Khi mua x cây bút sẽ được tặng y cây bút nữa.

**Yêu cầu:**

Hãy đưa ra số bút tối thiểu mà Thầy Ba phải mua để thầy có đủ s cây bút.

**Input**: 1 chuỗi 3 giá trị số nguyên bao gồm x, y và s. Các số cách nhau bởi dấu “,”

Điều kiện 0 < x < 109, 0 < y < 109, 0 < s < 109

**Output**: 1 giá trị số nguyên là số bút tối thiểu mà Thầy Ba mua.

**Ví dụ:**

* Với x = 2, y = 1, s = 6, thì promotion(x,y,s) = 4.

Giải thích: Thầy Ba sẽ mua 4 cây bút như sau:

Mua 2 cây bút được khuyến mãi 1 cây -> tổng cộng thầy có 3 cây bút.

Mua thêm 2 cây bút được khuyến mãi 1 cây -> tổng cộng thầy có 6 cây bút.

* Với x = 3, y = 2, s = 11, thì promotion(x,y,s) = 7.  
  Giải thích: Thầy Ba sẽ mua 7 cây bút như sau:

Mua 3 cây bút được khuyến mãi 2 cây -> tổng cộng thầy có 5 cây bút.

Mua thêm 3 cây bút được khuyến mãi 2 cây -> tổng cộng thầy có 10 cây bút.

Mua thêm 1 cây bút không cần khuyến mãi -> tổng cộng thầy có 11 cây bút.

**Test case:**

* **Input**: 2, 1, 6

**Output**: 4.

* **Input**: 3, 2, 11

**Output**: 7.

* **Input**: 3, 2, 5

**Output**: 3.

* **Input**: 4, 2, 11

**Output**: 8.

* **Input**: 5, 2, 10

**Output**: 8.

**Bài 12:**

**Level: Trung bình**

**Mô tả:**

Để chúc mừng ngày 8/3, các bạn học sinh lớp 12A3 dự định tổ chức dã ngoại đến biển Nha Trang và lớp sẽ đi bằng taxi.

Các bạn trong lớp được chia thành n nhóm, mỗi nhóm thứ i gồm x bạn (1 ≤ x ≤ 4 vì mỗi chiếc taxi chở tối đa 4 hành khách).

**Yêu cầu:**

Lớp 12A3 cần thuê ít nhất bao nhiêu chiếc taxi để chở các nhóm đi, với điều kiện là các bạn trong nhóm phải ngồi chung taxi (một taxi có thể chở một nhóm trở lên).

**Input:** 1 chuỗi chứa dãy số nguyên, mỗi số cách nhau bởi 1 dấu “,”

**Output**: Số lượng tối thiểu taxi cần thiết để chở tất cả học sinh đến nơi.

**Ví dụ:**

Với arr = [1 ,2 ,4 ,3 ,3] thì taxi(arr) = 4  Các bạn học sinh có thể sắp xếp như sau:  
- Nhóm thứ 3 gồm 4 học sinh  
- Nhóm thứ 4 gồm 3 học sinh  
- Nhóm thứ 5 gồm 3 học sinh  
- Nhóm thứ nhất và nhóm thứ 2 gồm 3 học sinh (1 và 2 học sinh) Vậy cần 4 taxi.

**Test case:**

* **Input**: 1 ,2 ,4 ,3 ,3

**Output**: 4

* **Input**: 1 ,2 ,4 ,2 ,3

**Output**: 3

* **Input**: 4 ,2 ,4 ,3 ,3

**Output**: 5

* **Input**: 1 ,2 ,4 ,3, 1, 2

**Output**: 4

* **Input**: 2 ,2 ,4 ,3, 2, 3

**Output**: 5

**Bài 13:**

**Level: Dễ**

**Yêu cầu:**

Viết phương thức trả về index số lớn thứ nhì trong mảng. Nếu không có số lớn nhì thì trả về -1. Nếu có nhiều số lớn nhì thì lấy vị trí của số đầu tiên.

**Input:** 1 chuỗi chứa dãy số nguyên, mỗi số cách nhau bởi 1 dấu “,”

**Output**: int

- Index của số lớn thứ 2.

**Ví dụ:** Cho mảng [8, 3, 6, 1, 5, 4], kết quả là 2. Vì 6 là số lớn thứ nhì, index của 6 là 2. Lưu ý nếu có 2 số lớn thứ 2 thì trả về index của số đầu tiên

**Ví dụ:**

* **Input**: 8, 3, 6, 1, 5, 4

**Output**: 2.

* **Input**: 8, 3, 4, 1, 5, 4

**Output**: 4.

* **Input**: 10, 3, 8, 1, 9, 4, 9

**Output**: 4.

* **Input**: 8, 8, 8, 8, 8

**Output**: -1.

* **Input**: 8, 3, 6, 1, 5, 4, 7

**Output**: 6.

**Bài 14:**

**Level: Dễ**

Viết phương thức đếm số ký tự khác nhau ở chuỗi đầu vào.

**Input**: String

- Chuỗi cần đếm số ký tự khác nhau.

**Output**: int

- Số lượng ký tự khác nhau.

**Ví dụ:** s = "cabca", thì kết quả là 3.

Bởi vì bao gồm 3 ký tự khác nhau a, b, c.

**Test case:**

* **Input**: cabca

**Output**: 3.

* **Input**: cabcde

**Output**: 5.

* **Input**: ccccccccccccc

**Output**: 1.

* **Input**: 12sdvcs

**Output**: 6.

* **Input**: ca

**Output**: 2.

**Bài 15:**

**Level: Trung bình**

**Yêu cầu:**

Viết phương thức để kiểm tra cách đặt tên ở Java, cho 4 loại: “package”, “class”, “field”, “method”.

- Tên package: snake\_case.

- Tên class: PascalCase.

- Tên field and method: camelCase.

**Input**: 2 tham số String.

- Dòng đầu tiên: Tên file cần kiểm tra.

- Dòng thứ 2: Loại file bao gồm 4 loại: “package”, “class”, “field”, “method” (không quan tâm kí tự hoa, thường).

**Output**: boolean

- true: nếu cách đặt tên hợp lệ. Ngược lại, trả về false.

**Ví dụ:**

* checkRuleName ("HelloWorld", "class") = true
* checkRuleName ("HelloWorld", "CLASS") = true
* checkRuleName ("HelloWorld", "cLAss") = true
* checkRuleName ("helloWorld", "cLAss") = false

**Test case:**

* **Input**

HelloWorld

class

**Output**: true

* **Input**:

HelloWorld

CLASS

**Output**: true

* **Input**:

HelloWorld

cLAss

**Output**: true

* **Input**:

helloWorld

cLAss

**Output**: false

* **Input**:

helloWorld

field

**Output**: true

**Bài 16:**

**Level: Dễ**

**Yêu cầu:**

Cho mảng a chứa các số nguyên

Xây dựng mảng b là prefix của mảng a cụ thể theo công thức:

b[0] = a[0]

b[1] = a[0] + a[1]

b[2] = a[0] + a[1] + a[2]

...

b[n - 1] = a[0] + a[1] + ... + a[n - 1]

với n là độ dài của mảng a.

**Input:** 1 chuỗi chứa dãy số nguyên, mỗi số cách nhau bởi 1 dấu “,”

**Output**: 1 mảng số nguyên

**Ví dụ**

Với a = [1, 2, 3], thì kết quả prefixSums(a) = [1, 3, 6].

Mảng b được tạo ra: [1, 1 + 2, 1 + 2 + 3] = [1, 3, 6]

**Test case**

* **Input**: 1, 2, 3

**Output**: [1, 3, 6].

* **Input**: 1, 4, 3

**Output**: [1, 5, 8].

* **Input**: 2, 1, 5

**Output**: [2, 3, 8].

* **Input**: 1, 5, 7

**Output**: [1, 6, 13].

* **Input**: 1, 2, 3, 4, 0

**Output**: [1, 3, 6, 10, 10].

**Bài 17:**

**Level: Trung bình**

**Yêu cầu**

Cho hai xâu ký tự, hãy kiểm tra xem có tồn tại phương án đổi chỗ các kí tự ở trong xâu thứ nhất để trở thành xâu thứ hai được hay không?

**Input**:

Dòng đầu tiên là xâu ký tự thứ nhất

Dòng thứ 2 là xâu ký tự thứ hai

**Output**: boolean

**Ví dụ**

* Với string1 = "abcd" và string2 = "cbad" thì kết quả charactersRearrangement(string1, string2) = true;
* Với string1 = "ab" và string2 = "bc" thì kết quả charactersRearrangement(string1, string2) = false.

**Test case:**

* **Input**:

abcd

cbad

**Output**: true

* **Input**:

ab

bc

**Output**: false.

* **Input**:

abcc

bcd

**Output**: false.

* **Input**:

abcc

bcac

**Output**: true.

* **Input**:

Ab123

23ba1

**Output**: false. ( hoa thường khác nhau)

**Bài 18:**

**Level: Dễ**

**Yêu cầu:**

Cho một mảng số nguyên

Hãy viết chương trình tìm ra một mảng mới, sao cho các phần tử trong 2 mảng này chứa đủ các số nguyên liên tiếp của một khoảng nào đó

**Input:** 1 chuỗi chứa dãy số nguyên, mỗi số cách nhau bởi 1 dấu “,”

**Output**: 1 mảng số nguyên

**Ví dụ:**

Với sequence = [6, 2, 3, 8], thì kết quả makeArrayConsecutive(sequence) = [4, 5, 7].

Với mảng mới gồm các số [4,5,7], thì hai mảng có chứa đủ các số nguyên liên tiếp từ 2 tới 8

**Test case:**

* **Input**: 6, 2, 3, 8

**Output**: [4, 5, 7].

* **Input**: 6, 2, 3, 8, 10

**Output**: [4, 5, 7, 9].

* **Input**: 6, 1, 3, 8, 11

**Output**: [2, 4, 5, 7, 9, 10].

* **Input**: 6, 11

**Output**: [7, 8, 9, 10].

* **Input**: 2, 3, 8

**Output**: [4, 5, 6, 7].

**Bài 19:**

**Level: Dễ**

Cho một mảng số nguyên

Hãy kiểm tra xem tần số xuất hiện của tất cả các số nguyên trong mảng có bằng nhau hay không

**Input:** 1 chuỗi chứa dãy số nguyên, mỗi số cách nhau bởi 1 dấu “,”

**Output**: 1 giá trị Boolean

**Ví dụ:**

* Với inputArray = [1, 2, 2, 1], thì kết quả checkEqualFrequency(inputArray) = true. Số 1 và số 2 đều có tần số xuất hiện là 2
* Với inputArray = [1, 2, 2, 3, 1, 3, 1, 3], thì kết quả checkEqualFrequency(inputArray) = false. Số 1 có tần số xuất hiện là 3, khác vs số 2 và số 3 có tần số xuất hiện là 2

**Test case:**

* **Input**: 1, 2, 2, 1

**Output**: true.

* **Input**: 1, 2, 2, 3, 1, 3, 1, 3

**Output**: false.

* **Input**: 1, 2, 2, 3, 1, 3, 1, 3, 0

**Output**: false.

* **Input**: 1, 2, 2, 3, 1, 3, 1, 3, 2

**Output**: true.

* **Input**: 1, 2, 2, 0, 1, 3, 0, 3

**Output**: true.

**Bài 20:**

**Level: Dễ**

**Yêu cầu:**

Cho một mảng các xâu kí tự, hãy sắp xếp mảng tăng dần theo độ dài của xâu kí tự Nếu hai xâu kí tự có cùng độ dài, thứ tự của chúng giữ nguyên như thứ tự trong mảng ban đầu

**Input:** 1 chuỗi chứa dãy các xâu ký tự, mỗi xâu ký tự cách nhau bởi 1 dấu “,”

**Output**: 1 mảng các chuỗi ký tự đã được sắp xếp.

**Ví dụ:**

● Với inputArray = ["abc", "", "aaa", "a", "zz"] thì kết quả sortByLength(inputArray) = ["", "a", "zz", "abc", "aaa"]

Mảng được sắp xếp theo thứ tự tăng của độ dài xâu. Xâu "abc" có cùng độ dài với xâu "aaa", nhưng do đứng trước "aaa" trong mảng ban đầu nên vẫn đứng trước "aaa" trong mảng kết quả

**Test case:**

* **Input**:

abc, , aaa, a, zz

**Output**: ["", "a", "zz", "abc", "aaa"]

* **Input**:

abc, b, aaa, a, zz

**Output**: [“b”, "a", "zz", "abc", "aaa"]

* **Input**:

abc, , aaa, a, zz, 123

**Output**: ["", "a", "zz", "abc", "aaa", “123”]

* **Input**:

abc, , aa, a, zz

**Output**: ["", "a", "aa", "zz", "abc"]

* **Input**:

abc, ,a

**Output**: ["", "a", "abc "]

**Bài 21:**

**Level: Dễ**

**Yêu cầu:**

Tìm chữ số khác không cuối cùng của n!(giai thừa)

**Input**: 1 số nguyên

**Output**: 1 số nguyên

**Ví dụ:**

* Với n = 5, kết quả lastDigitDiffZero(n) = 2. Vì 5! = 1 \* 2 \* 3 \* 4 \* 5 = 120.
* Với n = 6, kết quả lastDigitDiffZero(n) = 2. Vì 6! =1 \* 2 \* 3 \* 4 \* 5 \* 6 = 720.
* Với n = 10, kết quả lastDigitDiffZero(n) = 8. Vì 10! = 3628800.

**Test case:**

* **Input**: 5

**Output**: 2.

* **Input**: 6

**Output**: 2.

* **Input**: 10

**Output**: 8.

* **Input**: 8

**Output**: 2.

* **Input**: 12

**Output**: 6.

**Bài 22:**

**Level: Trung bình**

**Mô tả**

Một người bạn của bạn muốn thử thách bạn một trò chơi nhỏ. Anh ta đưa cho bạn một con số và muốn bạn tìm hai số nguyên tố có tổng bằng số đã cho.

**Yêu cầu:**

Viết một hàm để tìm xem liệu có giải pháp khả thi với con số được cho không. Có thể có rất nhiều cách kết hợp, vì vậy hãy trả về cặp số duy nhất có một số nhỏ nhất trong số các số tạo ra các kết hợp thoả mãn.

**Input**: 1 số n cần kiểm tra

**Output**: 1 mảng số nguyên

**Ví dụ**

* Với n = 4,  đầu ra là [2,2].
* Với n = 17,  không có kết quả nào thoả mãn, đầu ra là [].
* Với n = 24, có rất nhiều cặp số thoải mãn như [5,19], [7,17], [11, 13], đầu ra là [5, 19] vì số nhỏ nhất trong các cặp số là 5.

**Test case:**

* **Input**: 4

**Output**: [2,2].

* **Input**: 17

**Output**: [].

* **Input**: 24

**Output**: [5, 19]

* **Input**: 8

**Output**: [3, 5]

* **Input**: 12

**Output**: [5,7]

**Bài 23:**

**Level: Dễ**

**Mô tả:**

Trong toán học, một cấp số cộng là một dãy số thoả mãn điều kiện: hai phần tử liên tiếp nhau sai khác nhau một hằng số. Chẳng hạn, dãy số 3, 5, 7, 9, 11,... là một cấp số cộng với các phân tử liên tiếp sai khác nhau hằng số 2.

**Yêu cầu:**

Cho một mảng số nguyên. Hãy kiểm tra xem nó có tạo thành một cấp số cộng hay không?

**Input:** 1 chuỗi chứa dãy số nguyên, mỗi số cách nhau bởi 1 dấu “,”

**Output**: 1 giá trị boolean

**Ví dụ:**

Với sequence = [1, 4, 7], thì kết quả isArithmeticProgression(sequence) = true;

Với sequence = [2, 4, 7], thì kết quả isArithmeticProgression(sequence) = false

**Test case:**

* **Input**: 1, 4, 7

**Output**: true;

* **Input**: 2, 4, 7

**Output**: false;

* **Input**: 1, 5, 9, 13

**Output**: true;

* **Input**: 0, 4, 8 , 11

**Output**: false;

* **Input**: 1, 4, 7, 10

**Output**: true;

**Bài 24**

**Level: Dễ**

**Yêu cầu:**

Hãy liệt kê các giá trị trong mảng 1 chiều các số nguyên có chữ số đầu tiên là chữ số lẻ

**Input:** 1 chuỗi chứa dãy số nguyên, mỗi số cách nhau bởi 1 dấu “,”. Yêu cầu các số < 60000000000

**Output**: 1 mảng số nguyên ( kiểu Long) gồm các phần tử có chữ số đầu tiên là chữ số lẻ

**Ví dụ:**

* Với mảng số nguyên [13, 4456, 723424, 234234, 534234] thì kết quả nhậnđược là [13, 723424, 534234];

**Test case:**

* **Input**: 13, 4456, 723424, 234234, 534234

**Output**: [13, 723424, 534234];

* **Input**: 22343534, 42312312, 7235345

**Output**: [7235345]

* **Input**: 22343534, 42312312, 7235345, 15, 1354323

**Output**: [7235345, 15, 1354323]

* **Input**: 42343534, 12312312, 6235345, 54342321321

**Output**: [12312312, 54342321321]

* **Input**: 22343534, 42312312, 6235345

**Output**: []

**Bài 25**

**Level: Dễ**

**Yêu cầu:**

Hãy tìm giá trị trong mảng các số thực xa giá trị x nhất

**Input:**

Dòng đầu tiên là 1 chuỗi chứa dãy số thực, mỗi số cách nhau bởi 1 dấu “,”

Dòng thứ 2 là giá trị x

**Output**: 1 giá trị số thực xa x nhất

Ví dụ:

Với arr = [13.5, 4456.23423423, 723424.12, 234234.1, 534234.432] và x = 15.5 thì kết quả là 723424.12. Vì 723424.12 trừ 15.5 ra kết quả là lớn nhất

**Test case:**

* **Input**:

13.5, 4456.23423423, 723424.12, 234234.1, 534234.432

15.5

**Output**: 723424.12

* **Input**:

22343534.12, 42312312.13, 7235345.154

42312312.12

**Output**: 7235345.154

* **Input**:

22343534.12, 42312312.13, 7235345.154

0

**Output**: 7235345.154

* **Input**:

22343534.12, 42312312.13, 7235345.154

7235350.154

**Output**: 22343534.12

* **Input**:

22343534.12, 42312312.13, 7235345.154, 0, -5

42312312.12

**Output**: -5