**Problem Description**

* Given a sorted array **A** containing **N** integers both positive and negative.

You need to create another array containing the **squares of all the elements** in **A** and return it in non-decreasing order.

 Try to do this in **O(N)** time.

**Problem Constraints**

1 <= N <= 105.

-103 <= A[i] <= 103

**Input Format**

First and only argument is an integer array **A**.

**Output Format**

Return a integer array as described in the problem above.

**Example Input**

Input 1:

A = [-6, -3, -1, 2, 4, 5]

Input 2:

A = [-5, -4, -2, 0, 1]

**Example Output**

Output 1:

[1, 4, 9, 16, 25, 36]

Output 2:

[0, 1, 4, 16, 25]

**public** **class** **Solution** **{**

**public** ArrayList**<**Integer**>** **solve(**ArrayList**<**Integer**>** A**)**

**{**

ArrayList**<**Integer**>** R **=** **new** ArrayList**<>();**

**int** i **=** 0**,** j **=** A**.**size**()** **-** 1**;**

**while(**i**<=**j**)**

**{**

**if(**Math**.**abs**(**A**.**get**(**i**))** **>** Math**.**abs**(**A**.**get**(**j**)))**

**{**

R**.**add**(**A**.**get**(**i**)\***A**.**get**(**i**));**

i**++;**

**}**

**else**

**{**

R**.**add**(**A**.**get**(**j**)\***A**.**get**(**j**));**

j**--;**

**}**

**}**

Collections**.**reverse**(**R**);**

**return** R**;**

**}**

**}**

* Given an array of integers, sort the array into a wave like array and return it,  
  In other words, arrange the elements into a sequence such that a1 >= a2 <= a3 >= a4 <= a5.....

**Example**

Given [1, 2, 3, 4]

One possible answer : [2, 1, 4, 3]

Another possible answer : [4, 1, 3, 2]

**public** **class** **Solution** **{**

**public** ArrayList**<**Integer**>** **wave(**ArrayList**<**Integer**>** a**)** **{**

Collections**.**sort**(**a**);**

**for(int** i **=** 0**;** i **<** a**.**size**()** **-** 1**;** i **=** i **+** 2**)** **{**

**int** temp **=** a**.**get**(**i**);**

a**.**set**(**i**,** a**.**get**(**i **+** 1**));**

a**.**set**(**i**+**1**,** temp**);**

**}**

**return** a**;**

**}**

**}**

**Problem Description**

Given four positive integers **A, B, C, D**, determine if there’s a rectangle such that the lengths of its sides are A, B, C and D (in any order).

If any such rectangle exist return **1** else return 0.

**Problem Constraints**

1 <= A, B, C, D <= 100

**Input Format**

First argument is an interger **A**.

Second argument is an interger **B**.

Third argument is an interger **C**.

Fourth argument is an interger **D**.

**Output Format**

If any such rectangle exist whose sides are A, B, C, D in any orde then return **1** else return 0.

**Example Input**

Input 1:

A = 1

B = 1

C = 2

D = 2

Input 2:

A = 1

B = 2

C = 3

D = 4

Determine whether an integer is a palindrome. Do this without extra space.

A palindrome integer is an integer x for which reverse(x) = x where reverse(x) is x with its digit reversed.  
Negative numbers are not palindromic.

**Example :**

Input : 12121

Output : True

Input : 123

Output : False

**public** **class** **Solution** **{**

**public** **boolean** **isPalindrome(int** a**)** **{**

**if(**a **==** check**(**a**))**

**return** **true;**

**else**

**return** **false;**

**}**

**public** **int** **check(int** num**){**

**int** reverted **=** 0**;**

**while** **(**num **>** 0**)** **{**

reverted **=** reverted**\***10 **+** num**%**10**;**

num **/=** 10**;**

**}**

**return** reverted**;**

**}**

**}**

**Problem Description**

Given an integer **A**, return the number of trailing zeroes in A!.

**Note**: Your solution should be in logarithmic time complexity.

\*\*Problem Constraints\*\*

1 <= A <= 10000

\*\*Input Format\*\*

First and only argumment is integer A.

\*\*Output Format\*\*

Return an integer, the answer to the problem.

\*\*Example Input\*\*

Input 1:

A = 4

Input 2:

A = 5

\*\*Example Output\*\*

Output 1:

0

Output 2:

1

\*\*Example Explanation\*\*

Explanation 1:

4! = 24

Explanation 2:

5! = 120

**class** **Solution** {

**public:**

**int** trailingZeroes(**int** n) {

**int** sum **=** 0;

**while** (n **/** 5 **>** 0) {

sum **+=** (n **/** 5);

n **/=** 5;

}

**return** sum;

}

};

**Problem Description**

Given a sorted array **A** and a target value **B**, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

You may assume no duplicates in the array.

\*\*Problem Constraints\*\*

1 <= |A| <= 100000

1 <= B <= 109

public class Solution {

public int searchInsert(ArrayList<Integer> a, int b) {

}

}

**Problem Description**

Given a string **A** of parantheses ‘(‘ or ‘)’.

The task is to find minimum number of parentheses ‘(‘ or ‘)’ (at any positions) we must add to make the resulting parentheses string valid.

An string is valid if:

* Open brackets must be closed by the corresponding closing bracket.
* Open brackets must be closed in the correct order.

**Problem Constraints**

1 <= |A| <= 105

A[i] = '(' or A[i] = ')'

You are given a string **S**, and you have to find all the **amazing substrings** of **S**.

Amazing Substring is one that starts with a **vowel** (a, e, i, o, u, A, E, I, O, U).

Input

ABEC

Output

6

Explanation

Amazing substrings of given string are :

1. A

2. AB

3. ABE

4. ABEC

5. E

6. EC

Given the array of strings **A**,  
you need to find the longest string S which is the prefix of **ALL** the strings in the array.

Longest common prefix for a pair of strings **S1** and **S2** is the longest string **S** which is the prefix of both **S1**  
and **S2**.

For Example, longest common prefix of "abcdefgh" and "abcefgh" is "abc".

Write a function that takes an unsigned integer and returns the number of 1 bits it has.

Example:

The 32-bit integer 11 has binary representation

00000000000000000000000000001011

so the function should return 3.

Note that since Java does not have unsigned int, use long for Java

public int numSetBits(long a) {

int setBits=0;

while(a != 0) {

if(a%2 == 1)

setBits++;

a = a/2;

}

return setBits;

}