GRADED ASSIGNMENT 3

PROBLEM 1

The battle between negative and non-negative numbers is an ancient one. In the ongoing battle somehow the negative numbers got scattered all over the battle field. They need to re-unite again so as to gain maximum power and then attack the non-negative numbers. Can you help the negative numbers in achieving this ?

Write a function **void arrangeVector(vector<int> & V)** to compute the following task.

The function re-arranges the elements of the vector V such that all negative numbers in the array appear before all the non-negative integers.
Note that the vector need not be sorted, only the negative numbers must appear first.
You should not use a secondary array or vector to accomplish this task.

NOTE:-
You should **ONLY** write this function
Do not write a **MAIN FUNCTION.** If you do so, your code will not compile and will be awarded a zero.
Also, **do not change the name of the function** and fill in your code, only write in the function body provided.
Also, you do not need to take any input from the user and should not print anything. So, do not use any cin or cout statements.

INPUT FORMAT

- No input

CONSTRAINTS

- The length of the array varies from 10 to 10000.

OUTPUT FORMAT

- No output, just modify the vector in place to perform the given task

SAMPLE EXAMPLE:

If V = {-5, 7, 8, 9 , 12, -4, -7, 2, 4, 77, -88, 99, 10, 1, 0}
After rearrangement V = {-5, -88, -4, -7, 7, 8, 9, 10, 1, 77, 99, 12, 0, 2, 4}
Note that there can be multiple ways to represent the above. Like {-88, -5, -7, -4, 7, 8, 9, 10, 1, 77, 99, 12, 0, 2, 4} is also a correct and valid solution. You can do any one of them.

PROBLEM 2

Surinder is fond of candies. He has a bag full of candies containing different types of candies. He has plans to go for a picnic. Carrying too many candies would be difficult for him so he wants to take exactly one candy of each type that he possess. Can you help him out?

So here is the task. You are given function **void unique_elements(vector<int> & V).** Vector V contains integers denoting the type of each candy. You have to modify Vector V such that it contains only unique integers which would subsequently mean selecting exactly one candy of each type.

NOTE:-
You need to only implement the **unique_elements** function in your code.
Do not write the main method in your code.
Just modify the Vector V within **void unique_elements(vector<int> & V)** so that it contains the correct desired result.

INPUT FORMAT

- No input from standard console

CONSTRAINTS

- All values in Vector V are Integers

OUTPUT FORMAT

- No Output

SAMPLE EXAMPLE

Suppose initially Vector V contains { 6, 1, 2, 5, 1, 6, 2 }
Final Vector V should contain { 6, 1, 2, 5 }
The order of elements in Final Vector V can be any permutation of the contained values, i.e. { 6, 1, 2, 5 } = { 1, 2, 6, 5 } = { 1, 5, 6, 2 } = ...

PROBLEM 3

Playing with strings has been an old bad habit of computer scientists. Sometimes they splice, Sometimes they split and Sometimes they append. You being a computer scientist from IITD need to also play the GAME OF STRINGS now.

So here is the task.

You are given a function **string find_element(vector<string> & V)**
Vector V contains strings which are unique. Your function should return the 3rd last string when all the strings are compared in lexicographic order.

https://en.wikipedia.org/wiki/Lexicographical_order

NOTE:-
You need to only implement the **string find_element(vector<string> & V)** function in your code. Do not write the main method in your code.
The function should return a string.

INPUT FORMAT

- No input from standard console

CONSTRAINTS

•All values in Vector V are strings

•Size of Vector V is strictly greater than 2

•All strings would contain alpha-numeric characters

•Max Size of each string is 105

OUTPUT FORMAT

•No Output

EXAMPLE
Say V contains { "Ram", "Hari", "Shyam", "Kishan", "Gopal" }
Then the 3rd last string after re-arranging them in lexicographic order would be "Kishan"

PROBLEM 4

Pascal is really a crazy guy. He always tries to play with numbers. This time while playing, unknowingly he discovered an interesting pattern. But Pascal is very poor in coding. So he wants you to implement it which would subsequently help him in his further research.

So here is the task.

Write a function **vector<long long int> nthPascalRow(int n)** to compute the nth row of Pascal triangle.

https://en.wikipedia.org/wiki/Pascal%27s_triangle

You can use array or vectors to calculate the answer but should not use any formula.

NOTE:-

You should **ONLY** write this function
Do not write a **MAIN FUNCTION.** If you do so, your code will not compile and will be awarded a zero.
Also, **do not change the name of the function** and fill in your code, only in the function body provided.
Also, you do not need to take any input from the user and should not print anything. So, do not use any cin or cout statements.

INPUT FORMAT

- No input

CONSTRAINTS

- n varies from 1 to 50 (Both Inclusive)

OUTPUT FORMAT

- No output, just return a vector<long long int> that contains, the nth row of the Pascal's triangle.