ARRAY 1

Algorithm Problem Solving – Samsung Vietnam R&D Center

Compose by phuong.ndp

Agenda

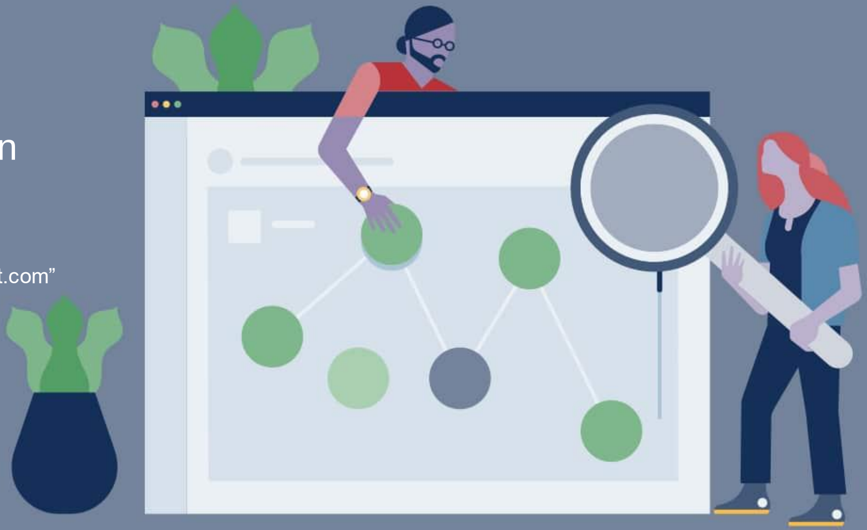
- Algorithm
- Array
- Bubble Sort
- Counting Sort
- Problem Solving

What is Algorithm?

- An algorithm is **a procedure or formula for solving a problem**, based on conducting a sequence of specified actions.

- A computer program can be viewed as an **elaborate algorithm**. In mathematics and computer science, an algorithm usually means **a small procedure that solves a recurrent problem**.

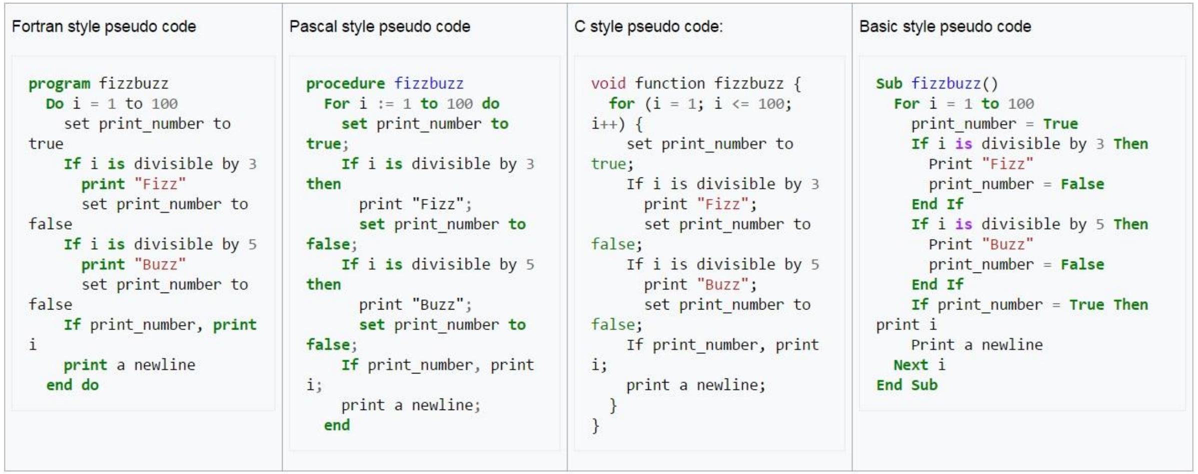
Quote "whatis.techtarget.com"



Represent algorithms

PSEUDO-CODE is a methodology that allows the programmer to represent the implementation of an algorithm.

A **FLOWCHART** is the graphical or pictorial representation of an algorithm with the help of different symbols, shapes and arrows in order to demonstrate a process or a program.



Fortran style pseudo code

```
program fizzbuzz
  Do i = 1 to 100
    set print_number to
true
    If i is divisible by 3
      print "Fizz"
      set print_number to
false
    If i is divisible by 5
      print "Buzz"
      set print_number to
false
    If print_number, print
i
    print a newline
  end do
```

Pascal style pseudo code

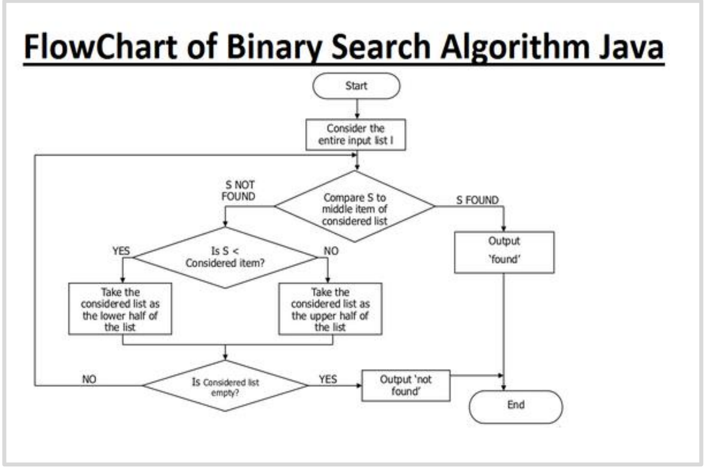
```
procedure fizzbuzz
  For i := 1 to 100 do
    set print_number to
true;
    If i is divisible by 3
then
      print "Fizz";
      set print_number to
false;
    If i is divisible by 5
then
      print "Buzz";
      set print_number to
false;
    If print_number, print
i;
    print a newline;
  end
```

C style pseudo code:

```
void function fizzbuzz {
  for (i = 1; i <= 100;
i++) {
    set print_number to
true;
    If i is divisible by 3
      print "Fizz";
      set print_number to
false;
    If i is divisible by 5
      print "Buzz";
      set print_number to
false;
    If print_number, print
i;
    print a newline;
  }
}
```

Basic style pseudo code

```
Sub fizzbuzz()
  For i = 1 to 100
    print_number = True
    If i is divisible by 3 Then
      Print "Fizz"
      print_number = False
    End If
    If i is divisible by 5 Then
      Print "Buzz"
      print_number = False
    End If
    If print_number = True Then
print i
    Print a newline
  Next i
End Sub
```


FlowChart of Binary Search Algorithm Java
Start
Consider the entire input list l
Compare S to middle item of considered list
S NOT FOUND
S FOUND
Output 'found'
Is S < Considered item?
YES
NO
Take the considered list as the lower half of the list
Take the considered list as the upper half of the list
Is Considered list empty?
NO
YES
Output 'not found'
End

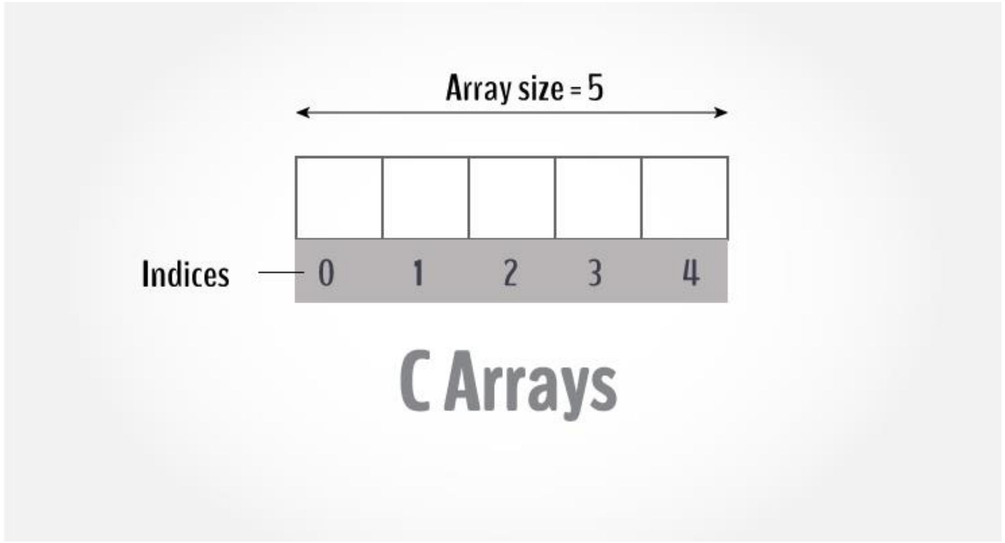
Array

What is an array?

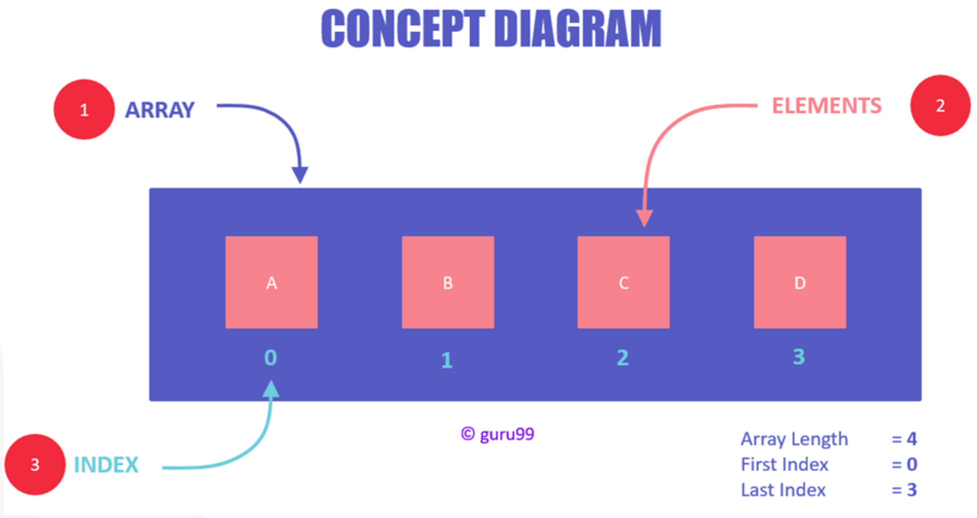
- An array is a data structure for storing more than one data item that has a similar data type. The items of an array are al located at adjacent memory locations. These memory locations are called **elements** of that array.
- The total number of elements in an array is called **length**. The details of an array are accessed about its position. This r eference is called **index** or **subscript**.

For example, if you want to store 5 integers, you ca n create an array for it.

int data[5];



What is an array?



The above diagram illustrates that:

1. An array is a container of elements.
2. Elements have a specific value and data type, like "ABC", TRUE or FALSE, etc.
3. Each element also has its own index, which is used to access the element.

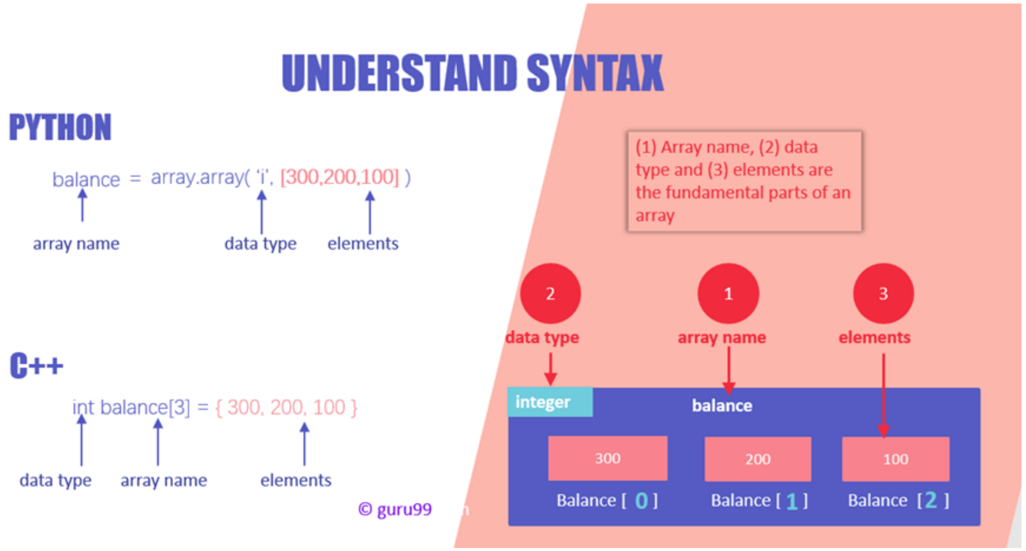
Note:

- Elements are stored at contiguous memory locations.
- An index is always less than the total number of array items.
- In terms of syntax, any variable that is declared as an array can store multiple values.
- Almost all languages have the same comprehension of arrays but have different ways of declaring and initializing them.
- However, three parts will always remain common in all the initializations, i.e., **array name**, **elements**, and the **data type of elements**.

What is an array?

- The following diagram illustrates the syntax of declaring an array in Python and C++ to present that the comprehension remains the same though the syntax may slightly vary in different languages.

- **Array name:** necessary for easy reference to the collection of elements
- **Data Type:** necessary for type checking and data integrity
- **Elements:** these are the data values present in a n array



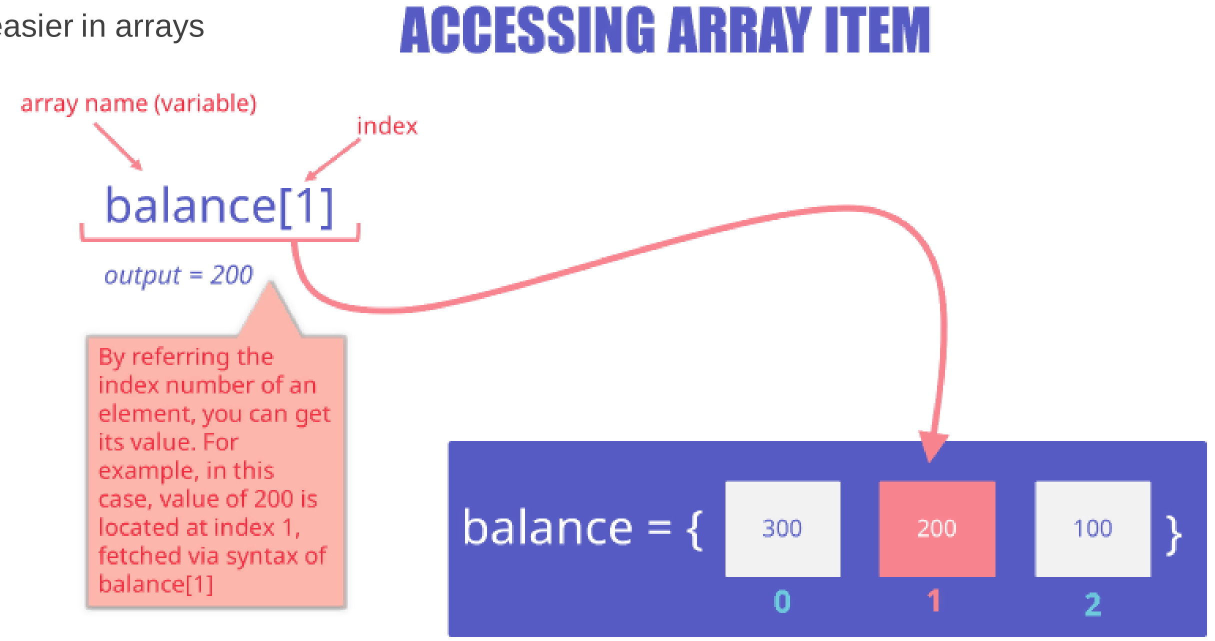
Why do we need arrays?

Here, are some reasons for using arrays:

- Arrays are best for storing multiple values in a single variable
- Arrays are better at processing many values easily and quickly
- Sorting and searching the values is easier in arrays

You can access any array item by using its index.

SYNTAX: arrayName[indexNum]
EXAMPLE: balance[1]



By referring the index number of an element, you can get its value. For example, in this case, value of 200 is located at index 1, fetched via syntax of balance[1]

Sorting Algorithms

What is sorting algorithm?

- A Sorting Algorithm is used to rearrange a given array or list elements according to a comparison operator on the elements. The comparison operator is used to decide the new order of element in the respective data structure.
- **For example**: The below list of characters is sorted in increasing order of their ASCII values. That is, the character with lesser ASCII value will be placed first than the character with higher ASCII value.
- p h u o n g n d p => d g h n n p p o u

Sorting Algorithms :

+ Selection Sort	+ Bubble Sort
+ Recursive Bubble Sort	+ Insertion Sort
+ Recursive Insertion Sort	+ Merge Sort
+ Iterative Merge Sort	+ Quick Sort
+ Iterative Quick Sort	+ Heap Sort
+ Counting Sort	+ Radix Sort
+ Bucket Sort	+ ShellSort
+ TimSort	+ Comb Sort…

Bubble Sort

- Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.

- **First Pass:**
(**5 1** 4 2 8) –> (**1 5** 4 2 8), Here, algorithm compares the first two elements, and swaps since 5 > 1.
(1 **5 4** 2 8) –> (1 **4 5** 2 8), Swap since 5 > 4
(1 4 **5 2** 8) –> (1 4 **2 5** 8), Swap since 5 > 2
(1 4 2 **5 8**) –> (1 4 2 **5 8**), Now, since these elements are alr eady in order (8 > 5), algorithm does not swap them.

- **Second Pass:**
(**1 4** 2 5 8) –> (**1 4** 2 5 8)
(1 **4 2** 5 8) –> (1 **2 4** 5 8), Swap since 4 > 2
(1 2 **4 5** 8) –> (1 2 **4 5** 8)
(1 2 4 **5 8**) –> (1 2 4 **5 8**)

- **Third Pass:**
(**1 2** 4 5 8) –> (**1 2** 4 5 8)
(1 **2 4** 5 8) –> (1 **2 4** 5 8)
(1 2 **4 5** 8) –> (1 2 **4 5** 8)
(1 2 4 **5 8**) –> (1 2 4 **5 8**)

- Now, the array is already sorted, but our algorithm does not know if it is completed. The algorithm needs one **wh ole** pass without **any** swap to know it is sorted.

Problem Solving

The task is to complete bubble() function which is used to implement Bubble Sort!

Example 1:

- **Input**: N = 5, arr[] = { 4, 1, 3, 9, 7}
- **Output**: 1 3 4 7 9

Example 2:

- **Input**: N = 10, arr[] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1}
- **Output**: 1 2 3 4 5 6 7 8 9 10

Your Task: This is a **function** problem. You only need to complete the **function bubble() that sorts** the array. Printing is done **automatically** by the **driver code**.

Expected Time Complexity: O(N).
Expected Auxiliary Space: O(1).

Constraints:
1 <= N <= 10^3
1 <= arr[i] <= 10^3

Counting Sort

- Counting sort is a sorting technique based on keys between a specific range. It works by counting the number of objects having distinct key values (kind of hashing). Then doing some arithmetic to calculate the position of each object in the output sequence.

For simplicity, consider the data in the range 0 to 9.

Input data: 1, 4, 1, 2, 7, 5, 2

1) Take a count array to store the count of each unique object.

Index:	0	1	2	3	4	5	6	7	8	9
Count:	0	2	2	0	1	1	0	1	0	0

Count[1] = frequency of 1 … Count[i] = frequency of i

2) Modify the count array such that each element at each index stores the sum of previous counts.

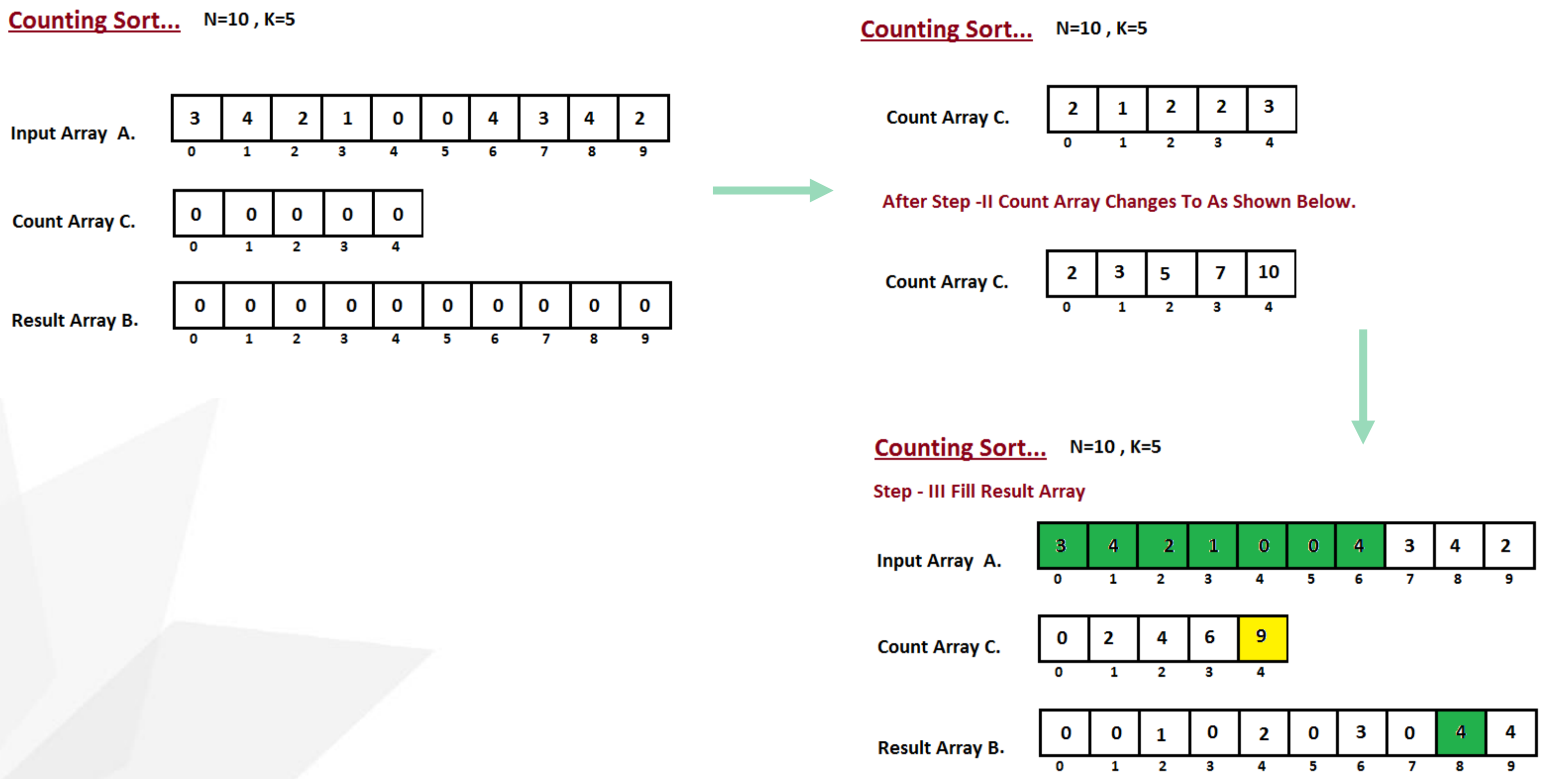
from i=1; Count[i] = Count[i-1] + Count[i]

Index:	0	1	2	3	4	5	6	7	8	9
Old Count:	0	2	2	0	1	1	0	1	0	0
New Count:	0	2	4	4	5	6	6	7	7	7

3) Output each object from the input sequence followed by decreasing its count by 1.
Process the input data: 1, 4, 1, 2, 7, 5, 2. Position of 1 is 2.
Put data 1 at index 2 in output. Decrease count by 1 to place next data 1 at an index 1 smaller than this index.

- The modified count array indicates the position of each object in the output sequence.

Counting Sort



Counting Sort... N=10 , K=5

Input Array A.

0	1	2	3	4	5	6	7	8	9
3	4	2	1	0	0	4	3	4	2

Count Array C.

0	1	2	3	4
0	0	0	0	0

Result Array B.

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0

Counting Sort... N=10 , K=5

Count Array C.

0	1	2	3	4
2	1	2	2	3

After Step -II Count Array Changes To As Shown Below.

Count Array C.

0	1	2	3	4
2	3	5	7	10

Counting Sort... N=10 , K=5

Step - III Fill Result Array

Input Array A.

0	1	2	3	4	5	6	7	8	9
3	4	2	1	0	0	4	3	4	2

Count Array C.

0	1	2	3	4
0	2	4	6	9

Result Array B.

0	1	2	3	4	5	6	7	8	9
0	0	1	0	2	0	3	0	4	4

Problem Solving

Find a number with **highest appearance frequency** among the given numbers

In the example below, the result will be 8

10, 8, 7, 2, 2, 3, 8, 8, 9, 8, 5, 3, 5

Problem Solving

As usual Babul is back with his problem but this time with numbers. In his problem there is a number **P**(always a whole number) with **N** digits. Now he started finding out the largest possible even number formed by rearranging this N digit number. For example consider number 1324, after rearranging the digits the largest even number possible is 4312.

Note: In case the number does not contain any even digit then output the largest odd number possible.

Input:

The first line of input will contain an integer **T** which is the number of testcases. Each of the next T lines will contain a number P.

Output:

For each test case in a new line print the required result.

Constraints: $1 <= T <= 100$ and $1 <= N <= 10^7$

Explanation:

Testcase 1: The largest even number formed will be: 4312.

Input	Output
5	
1324	4312
3415436	6543314
1023422	4322210
03517	75310
3555	5553

Thank you!

Algorithm Problem Solving – Samsung Vietnam R&D Center

Compose by phuong.ndp

Source

https://whatis.techtarget.com/definition/algorithm

https://www.edrawsoft.com/explain-algorithm-flowchart.html

https://www.guru99.com/array-data-structure.html

https://www.programiz.com/c-programming/c-arrays

https://www.geeksforgeeks.org/bubble-sort/

https://nguyenvanhieu.vn/counting-sort/

https://www.geeksforgeeks.org/counting-sort/

https://practice.geeksforgeeks.org/problems/largest-even-number/0