Arrange given numbers to form the biggest number

Given an array of numbers, arrange them in a way that yields the largest value. For example, if the given numbers are {54, 546, 548, 60}, the arrangement 6054854654 gives the largest value. And if the given numbers are {1, 34, 3, 98, 9, 76, 45, 4}, then the arrangement 998764543431 gives the largest value.

A simple solution that comes to our mind is to sort all numbers in descending order, but simply sorting doesn't work. For example, 548 is greater than 60, but in output 60 comes before 548. As a second example, 98 is greater than 9, but 9 comes before 98 in output.

So how do we go about it? The idea is to use any comparison based sorting algorithm. In the used sorting algorithm, instead of using the default comparison, write a comparison function myCompare() and use it to sort numbers. Given two numbers X and Y, how should myCompare() decide which number to put first – we compare two numbers XY (Y appended at the end of X) and YX (X appended at the end of Y). If XY is larger, then X should come before Y in output, else Y should come before. For example, let X and Y be 542 and 60. To compare X and Y, we compare 54260 and 60542. Since 60542 is greater than 54260, we put Y first.

Following is C++ implementation of the above approach. To keep the code simple, numbers are considered as strings, and vector is used instead of normal array.

```
// Given an array of numbers, program to arrange the numbers to form the
// largest number
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
using namespace std;

// A comparison function which is used by sort() in printLargest()
int myCompare(string X, string Y)
{
    // first append Y at the end of X
    string XY = X.append(Y);

    // then append X at the end of Y
    string YX = Y.append(X);

    // Now see which of the two formed numbers is greater
    return XY.compare(YX) > 0 ? 1: 0;
}

// The main function that prints the arrangement with the largest value.
```

```
// The function accepts a vector of strings
void printLargest(vector<string> arr)
{
    // Sort the numbers using library sort funtion. The function uses
    // our comparison function myCompare() to compare two strings.
    // See http://www.cplusplus.com/reference/algorithm/sort/ for details
    sort(arr.begin(), arr.end(), myCompare);

    for (int i=0; i < arr.size() ; i++ )
        cout << arr[i];
}

// driverr program to test above functions
int main()
{
    vector<string> arr;

    //output should be 6054854654
    arr.push_back("54");
    arr.push_back("546");
    arr.push_back("548");
    arr.push_back("60");
    printLargest(arr);

    // output should be 777776
    /*arr.push_back("7");
    arr.push_back("776");
    arr.push_back("7");
    arr.push_back("7");*/

    //output should be 998764543431
    /*arr.push_back("1");
    arr.push_back("34");
    arr.push_back("3");
    arr.push_back("98");
    arr.push_back("9");
    arr.push_back("76");
    arr.push_back("45");
    arr.push_back("4");
    */

   return 0;
}
```

Run on IDE

Output:

```
6054854654
```

This article is compiled by **Ravi Chandra Enaganti**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

79 Comments Category: Arrays

Related Posts:

- Longest Span with same Sum in two Binary arrays
- Count Inversions of size three in a give array
- Find the subarray with least average
- Count triplets with sum smaller than a given value
- Find zeroes to be flipped so that number of consecutive 1's is maximized
- Reorder an array according to given indexes
- Find maximum value of Sum(i*arr[i]) with only rotations on given array allowed
- Find maximum average subarray of k length

(Login to Rate and Mark)

2.6 Average Difficulty : **2.6/5.0**
Based on **11** vote(s)

Add to TODO List

Mark as DONE

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.