NATIONAL UNIVERSITY
OF COMPUTER & EMERGING SCIENCES
PESHAWAR CAMPUS

Problem Set:	Assignment: AG12	**Semester**:	Fall 2017
Points:	*See autograder*		
Date Set:	*See autograder*	**Due Date**:	*See autograder*
Course:	CS101 Introduction to Computing	**Instructor**:	Dr. Nauman

1 Big Integers

Since you are reading this, you have already downloaded and extracted the zip file.

1.1 Tasks to do

1. Make sure you have your editor/compiler of choice installed on your system. (If you are Windows, also make sure you have Dev-CPP installed. This is required to run local tests.)

2. You know that there is a limitation on the size of numbers we can put in any built-in datatype in C/C++. What happens if we want to manipulate numbers which are longer than this length?

   For example, if we want to add two numbers each of which is 60 digits long, we cannot do it with an int datatype or even with a long. So, how do we do it?

   One basic approach is to represent each number in an element of an array. Since an array can be of arbitrary length, we can store numbers of any length. However, since these are not "numbers" any more, we have to implement our own operations such as add, subtract etc. In this assignment, you will be writing a function that adds two such arrays.

   This function is declared as below:

   ```
   void array_sum(int a[], int b[], int c[], int size);
   ```

   This means that it takes in three arrays as its arguments as well as the maximum size of the arrays (since it's not possible to deduce the size of an array programmatically).

   Let's take an example of two numbers.

   ```
   num_a = 128
   num_b = 985
   ```

   When we convert them to arrays, they look like this:

   ```
   a = 8 2 1 0 0 0
   b = 5 8 9 0 0 0
   ```

   Notice that we have the least significant digit on the left-most side which is the opposite of how we think. This just makes the code a little easier to write.

   To add these two numbers, we simply add the first element of each array (`a` and `b`) and then store the result in the corresponding element of another array – called `c` in this function. The only problem is that if the result is greater than 10, we have to store the remainder in the corresponding element of `c` and carry the 1 forward. We keep doing this until all digits in both input arrays are done.

   Since we are working with arrays, we will be modifying the array that is passed to our function. Once we modify `c` in our function, we simply return (nothing) from the function and the caller will have the result in the array they passed to us. In essence, you only need to write code for adding a and b and storing the result in the array `c`.

   For our example above, by the end of your function, the value of `c` should be:

```
c = 3 1 1 1 0 0
```

since 128+985 = 1113

Your function should be able to handle very long numbers – up to 90+ digits.

3. To help you test your code, we've written a helper function called `main_test`. When you are writing your code, you can change the name of `main_test` to `main` and the name of existing `main` to `main_test`. You can also change the 128 and 985 in this function to help you debug your code.

   However, make sure you bring the names back to their original ones before you run local tests or submit the assignment.

4. That's it! Run your tests locally and if everything is good, submit to remote.