Due: Tuesday, November 13, 2012 by the beginning of class.

Honor Code: Like all work for this class, the Emory Honor Code applies. You should do your own work on all problems, unless you are explicitly instructed otherwise. If you get stuck or have questions, ask your instructor or a TA for help. For EACH of the files you submit, be sure to put the appropriate honor code statement (as specified on the course syllabus) at the top of the file in comments.

Preparation:

1. Create a directory called hw7 inside your cs170 project directory to save your hw6 files. 
   mkdir ~/cs170/hw7
2. Copy the starter files:
   cp ~cs170003/share/hw7/*.java ~/cs170/hw7
3. You must use ~/cs170/hw7 directory as your current directory when editing any program files for hw6. Change your current directory to your newly created hw7 directory:
   cd ~/cs170/hw7
4. You should see one file in your directory: CheckCC.java

Skills: In this homework, you will:

1. Practice working with single dimensional arrays, the modulo operator, and simple mathematical operators.
2. Break down large problems into smaller problems
3. Develop skills to test solutions to large problems in an incremental way.
4. Implement real-world algorithms using step-wise refinement

Background:

Most e-commerce websites these days take credit cards. Users must enter their credit card number and the merchant verifies that the number is valid. Then Visa, Mastercard, or AmEx process the payment to the merchant and pass the bill along to the user. However, users often mistype their credit card number by one or two digits. These common errors are why credit cards are designed with a secret. Using just the credit card number, we can detect most mistakes and errors caused by user mistakes/mistypes. The credit card number contains an error control code called a “checksum”. Specifically, the credit card number is formatted to comply with a Luhn-10 checking algorithm. In this homework, you will be writing code to verify a credit card number conforms to this algorithm. If it does not, you will specify what digit should be added to the number to make it valid.

Luhn-10 Algorithm:

The Luhn-10 algorithm is a weighted algorithm. Each digit in the credit card number is multiplied by a weight. These weights are then summed, forming the checksum. The checksum is divided by 10. If the remainder is 0, the credit card number is valid. If the remainder is NOT 0, the user made an error and can be prompted to re-enter their credit card data. The weighting for the Luhn-10 algorithm is as follows:

- Beginning with the first (ie leftmost) digit in the credit card, every other number is multiplied by 2.
If the product results in a 2 digit number (eg 6 x 2 = 12) then the individual digits (eg 1 and 2) are added to the checksum.

• The remaining digits of the credit card number are simply added to the checksum. That is, their weight is 1.

• Several (small) examples are given below, but this algorithm will work with your Visa or Mastercard number. Try it!:

**Example Number 1: 456392**

digit: 4 5 6 3 9 2
multiplied by: 2 1 2 1 2 1

| 8 | 5 | 12 | 3 | 18 | 2 |

checksum: 8 + 5 + 1+2 +3 + 1+8+ 2 = 30

Conclusion: This is a valid number since 30 % 10 == 0!

**Example Number 2: 4991657**

digit: 4 9 9 1 6 5 7
multiplied by: 2 1 2 1 2 1 2

| 8 | 9 | 18 | 1 | 12 | 5 | 14 |

checksum: 8 + 9 +1+8 + 1 + 1+2 +5 + 1+4 = 40

Conclusion: This is a valid number since 40 % 10 == 0

**Problems:**

For this homework, you will write two functions: one method which determines the checksum of a given array of integers, and a method which determines the final digit needed to make a given array of integers a valid credit card number. **DO NOT** edit the main method or the toArray method; you should only write the two methods below. Compile and run the program before attempting to write the methods below. The program will compile and run (although it will incorrectly tell you that any number you enter is valid).

1. **The method checksum**

   This method should determine what the checksum (ie the weighted sum) of a given number is. In the program you are given, CheckCC.java, examine the main method. When the user enters a number, it is transformed to an array of integers. This array of integers is then passed to the checksum method. Complete the method checksum so that it computes the weighted sum of the input parameter array of integers and returns the sum.

2. **The method generateDigit**

   Suppose you are given an arbitrary array of numbers. If the number is not a valid number, it could have a single digit appended to the end of it in order to make it valid. This function should determine what that digit should be. You should use the checksum function you wrote in #1 above to help you determine the digit. If a number is already a valid number, your method should return 0. Appending 0 to a valid number will not change its validity.
Here are some examples of determining the needed digit.

- **Number: 35672**
  Compute checksum as above in #1: 25 (therefore, this is not a valid number)
  This means we need to add 5 to our sum to make it an even 30 and a valid number.
  A number appended to the number 35672 will be multiplied by 1 due to its position. So we can simply append a 5 (making the number 356725) in order to make it valid.

- **Number: 3567**
  Compute checksum as above in #1: 21 (therefore, this is not a valid number)
  This means we need to add 9 to our sum to make it an even 30 and a valid number.
  A number appended to the number 3567 will be multiplied by 2.
  - There is no integer we can multiply by 2 and get 9 (this is the definition of even numbers).
  - Therefore, we have to generate 9 by adding two digits instead (ie 1+8).
  - We must generate multiply our digit by 2 and get 18.
  - The digit we need to append to our original number is 9. 35679 is a valid number.

**Grading:** Your grade will be determined based on:
- The correctness of your program and each method outlined above.
- Program style. Program style includes such things as comments, variable/method names, and readability.
- Late penalties apply as per the class syllabus.

**Submission:**
- Be sure to include the Honor Code statement at the top of EACH file you submit. Failure to do so will result in a 0 for that portion of the assignment.
- Only your last submission will be graded. Scores will be assigned based on the last submission only.
- Submit your work using the following commands. You need to be in your ~/cs170/hw7 directory when you issue them.
  - /home/cs170003/turnin-hw CheckCC.java hw7
- Your homework is not turned unless the above commands are successful (you will get a "success" message when turn in was successful).