AP Computer Science Name: ______________________________
Chapter 5 Programs

1. A pentagonal number is defined as $n(3n-1)/2$ for $n = 1, 2, ...,$ and so on. Therefore, the first few numbers are 1, 5, 12, 22, ... Write a method with following header that returns a pentgonal number.

```
public static int getPentagonalNumber(int n)
```

Write a test program that uses this method to display the first 100 pentagonal numbers with 10 numbers with 10 numbers on each line. (10 points)

2. Write a method that computes the sum of the digits in an integer. Use the following method header:

```
public static int sumDigits(int n)
```

For example, `sumDigits(234)` returns `9`. (Hint: Use the `%` operator to extract digits, and the `/` operator to remove the extracted digit.) Use a loop to repeatedly extract and remove the digit until all the digits are extracted. Write a test program that prompts the user to enter an integer and displays the sum of all its digits. (10 points)

3. Write a method with the following header to display an integer in reverse order: (10 points)

```
public static void reverse(int number)
```

For example, `reverse(3456)` display `6543`. Write a test program that prompts the user to enter an integer and displays its reversal.

4. Write a method to compute the following series: (10 points)

$$m(i) = \frac{1}{2} + \frac{2}{3} + ... + \frac{i}{i+1}$$

Write a test program that displays the following table:

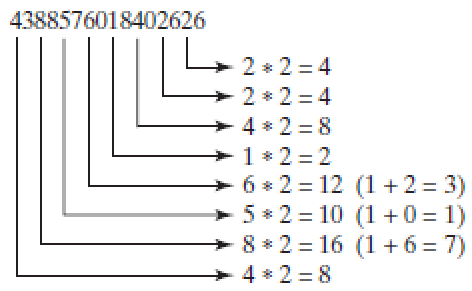
```
i                m(i)
_____________________

1                0.5000
2                1.1667
...
19               16.4023
20               17.3546
```

5. (*Financial: credit card number validation*) Credit card numbers follow certain patterns. A credit card number must have between 13 and 16 digits. It must start with: (30 points)

- 4 for Visa cards
- 5 for Master cards
- 37 for American Express cards
- 6 for Discover cards

In 1954, Hans Luhn of IBM proposed an algorithm for validating credit card numbers. The algorithm is useful to determine whether a card number is entered correctly or whether a credit card is scanned correctly by a scanner. Credit card numbers are generated following this validity check, commonly known as the *Luhn check* or the *Mod 10 check*, which can be described as follows (for illustration, consider the card number 4388576018402626):

1. Double every second digit from right to left. If doubling of a digit results in a two-digit number, add up the two digits to get a single-digit number.



2. Now add all single-digit numbers from Step 1.

$$4 + 4 + 8 + 2 + 3 + 1 + 7 + 8 = 37$$

3. Add all digits in the odd places from right to left in the card number.

$$6 + 6 + 0 + 8 + 0 + 7 + 8 + 3 = 38$$

4. Sum the results from Step 2 and Step 3.

$$37 + 38 = 75$$

5. If the result from Step 4 is divisible by 10, the card number is valid; otherwise, it is invalid. For example, the number 4388576018402626 is invalid, but the number 4388576018410707 is valid.

Write a program that prompts the user to enter a credit card number as a `long` integer. Display whether the number is valid or invalid. Design your program to use the following methods:

```java
/** Return true if the card number is valid */
public static boolean isValid(long number)

/** Get the result from Step 2 */
public static int sumOfDoubleEvenPlace(long number)

/** Return this number if it is a single digit, otherwise,
 * return the sum of the two digits */
public static int getDigit(int number)

/** Return sum of odd-place digits in number */
public static int sumOfOddPlace(long number)

/** Return true if the digit d is a prefix for number */
public static boolean prefixMatched(long number, int d)

/** Return the number of digits in d */
public static int getSize(long d)

/** Return the first k number of digits from number. If the
 * number of digits in number is less than k, return number. */
public static long getPrefix(long number, int k)
```

Here are sample runs of the program:

```
Enter a credit card number as a long integer:
  4388576018410707
4388576018410707 is valid
```

```
Enter a credit card number as a long integer:
  4388576018402626
4388576018402626 is invalid
```