Recursion Practice (Created by professor Kiyoung Song with modification)

Note: Answers can be found in ~ctse/cs211/lab25.tar on mars server

1) Write a recursive function named **reverse** that prints a positive integer backwards.
reverse(1234) → prints 4321

```
void reverse(int n) {
        …
}
```

2) The Fibonacci numbers is an integer sequence where the next number in the sequence is determined by the sum of the previous two numbers. The sequence starts off as the following:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144…

where:

1 + 1 = 2
1 + 2 = 3
2 + 3 = 5
3 + 5 = 8 … etc.

Create a recursive method that calculates the n^{th} number in the sequence.

fibonacci(3) = 2
fibonacci(4) = 3
fibonacci(5) = 5
fibonacci(6) = 8

Hint: For the base case, when n = 1 or n = 2, return 1.

```
int fibonacci(int n) {
        …
}
```

3) This is a classic problem and this particular wording is a variation on the text from Cracking the Coding Interview (question 8.11):

For parts a and b, you can first think about how to solve them iteratively without recursion. They are there to help you think about how to solve part c with recursion.

a) Given enough pennies (1 cent) and nickels (5 cents), how many ways can you make change for a given amount of cents? Your function prototype will be int ways(int cents).

Example: ways(12) will return 3.
(There are 3 ways: 2 nickels and 2 pennies, 1 nickel and 7 pennies, 12 pennies.)

b) Given enough pennies (1 cent), nickels (5 cents) and dimes (10 cents), how many ways can you make change?

c) Given enough pennies (1 cent), nickels (5 cents), dimes (10 cents) and quarters (25 cents), how many ways can you make change?

```
int ways(int cents, int coinType)
```

And you will call it this way:

```
cout << ways(100, 25);
>> 242
```