Problem 1. (*Sum of Integers*) Implement the functions `sum_iter()` and `sum_rec()` in `sum_of_ints.py` that take an integer n as argument and return the sum $S(n) = 1+2+3+\cdots+n$, computed iteratively (using a loop) and recursively. The recurrence relation for the latter implementation is

$$S(n) = \begin{cases} 1 & \text{if } n = 1, \\ n + S(n-1) & \text{if } n > 1. \end{cases}$$

```
$ python sum_of_ints.py 100
5050
5050
```

Problem 2. (*Exponentiation*) Implement the function `power()` in `power.py` that takes two integer arguments a and b and returns the value of a^b, computed recursively using the recurrence relation

$$a^b = \begin{cases} 1 & \text{if } b = 0, \\ aa^{b-1} & \text{if } b \text{ is odd}, \\ (a^2)^{b/2} & \text{if } b \text{ is even}. \end{cases}$$

```
$ python power.py 3 5
243
```

Problem 3. (*Bit Counts*) Implement the functions `zeros()` and `ones()` in `bits.py` that takes a bit string (ie, a string of zeros and ones) s as argument and returns the number of zeros and ones in s, each computed recursively. The number of zeros in a bit string is 1 or 0 (if the first character is 0 or 1) plus the number of zeros in the rest of the string; the empty string has 0 zeros. Similarly for the number of ones.

```
$ python bits.py 1010010010011110001011111
zeros = 11, ones = 14, total = 25
```

Problem 4. (*String Reversal*) Implement the function `reverse()` in `reverse.py` that takes a string s as argument and returns the reverse of the string, constructed recursively. The reverse of a string is the last character concatenated with the reverse of the string up to the last character; the reverse of an empty string is an empty string.

```
$ python reverse.py bolton
notlob
$ python reverse.py amanaplanacanalpanama
amanaplanacanalpanama
```

Problem 5. (*Palindrome*) Implement the function `is_palindrome()` in `palindrome.py`, using recursion, such that it returns `True` if the argument s is a palindrome (ie, reads the same forwards and backwards), and `False` otherwise. You may assume that s is all lower case and doesn't any whitespace characters. A string is a palindrome if the first character is the same as the last, and the rest of the string is a palindrome; an empty string is a palindrome.

```
$ python palindrome.py bolton
False
$ python palindrome.py amanaplanacanalpanama
True
```

Files to Submit

1. `sum_of_ints.py`
2. `power.py`
3. `bits.py`

4. `reverse.py`
5. `palindrome.py`

Before you submit:

- Make sure your programs meet the input and output specifications by running the following command on the terminal:

  ```
  $ python run_tests.py [<problems>]
  ```

  where the optional argument `<problems>` lists the numbers of the problems you want to test; all the problems are tested if no argument is given.