Recursion

Reading assignment:

What is Recursion: http://www.sparknotes.com/cs/recursion/whatisrecursion/
Examples of Recursion: http://www.sparknotes.com/cs/recursion/examples/

---

1. The Math class provides the method **pow(double a, double b)** that returns the value of the first argument raised to the power of the second argument.

   Write your own recursive method **power(double x, int n)** to compute x^n, where n is a non-negative integer.

   For example, power(3.0, 4) = 3.0 × 3.0 × 3.0 × 3.0 = 81.0; and power(2.5, 3) = 2.5 × 2.5 × 2.5 = 15.625. Test out your power() method and compare the results with the pow() method.

2. Trace the following recursive method using various input data. Restrict your data to non-negative integers. From your results, describe in words what does the method `m()` do.

```
// Precondition: n >= 0
public static int m(int n) {
   if (n == 0)
      return n;
   else
      return n%10 + m(n/10);
}
```

3. Trace the following recursive methods using various input data. Restrict your data to non-negative integers. From your results, describe in words what do the methods `P()` and `R()` do. If you interchange the two lines in method `P()` and call it `P1()`, what does the revised method `P1()` do?

   a.

```
// Precondition: n >= 0
public static void P(int n) {
   if (n < 10)
      System.out.print(n);
   else {
      P(n/10);                    // what if you swap
      System.out.print(n % 10);   // these two lines?
   }
}
```

   b.

```
// Precondition: n >= 0
public static void R(int n) {
   System.out.print(n % 10);
   if (n/10 != 0)
      R(n/10);
}
```

4. Given the following method `Q()`.

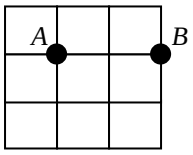
```
public static void Q(int n1, int n2) {
  if (n2 <= 0)
    System.out.println();
  else {
    Q(n1 – 1, n2 - 1);
    System.out.print(n1 + " ");
    Q(n1 + 1, n2 – 1);
  }
}
```

   a. What output is produced by the call **Q(14, 3)**? [*Hint:* First try Q(14,1), then Q(14,2)].

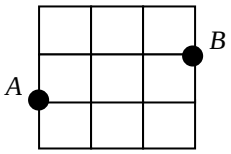
   b. If the `System.out.print(n1 + " ")` statement is moved before the first recursive call to Q(), what output will be produced by **Q(14,3)**?

   Write out a complete program to test out this method to check that the actual outputs match your answers above.

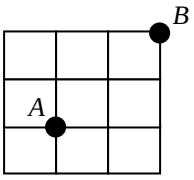
5. In a special town where you are only allowed to walk northwards or eastwards, each of the following examples shows the total number of unique north-east paths **ne(x, y)** to get from point *A* to point *B*, where *B* is *x* rows north and *y* columns east of *A*. Assume that *x* and *y* are non-negative integers. By convention, **ne(0, 0)** is 1.



ne(0, 2) = 1



ne(1, 3) = 4



ne(2, 2) = 6

   Write a recursive method to compute the number of north-east paths.

6. Study the code below and if you like, trace the recursive method **mystery()**. What is the smaller version of the task on which the recursive call works? How does the original problem relate to the smaller problem? What does the method compute?

```
public static void main(String[] args) {

  // code to read values into list - omitted

  System.out.println("Answer = " +
              mystery(list, list.length));
}
```

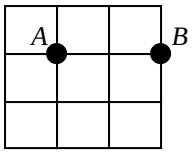
```
// Precondition: n > 0
public static int mystery(int[] arr, int n) {
  if (n == 1)
    return arr[0];
  else {
    int m = mystery(arr, n-1);
    return arr[n-1] > m ? arr[n-1] : m;
  }
}
```

7. Is the code in question 6 a "going-up", "going-down" or "split-half" recursion? Write the other two versions. You may also try any other version.

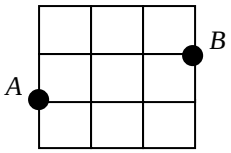
8. [Challenger]
   In question 5 above, we discussed the north-east paths problem. Now, modify the method so that besides reporting the number of possible paths, it also lists out all the possible paths, using 'N' to denote a step to the north and 'E' a step to the East.

   (*Hint:* You may represent the path taken so far as a string. Pass this string into the recursive call.)

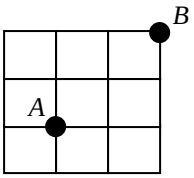
   You are allowed to violate the principle of cohesion for the sake of convenience here, that is, you may print the paths and return the number of paths in your method.



ne(0,2) = 1



ne(1,3) = 4



ne(2,2) = 6

The following are some sample outputs.

```
Enter x: 0
Enter y: 2
E E
Number of paths = 1
```

```
Enter x: 1
Enter y: 3
N E E E
E N E E
E E N E
E E E N
Number of paths = 4
```

```
Enter x: 2
Enter y: 2
N N E E
N E N E
N E E N
E N N E
E N E N
E E N N
Number of paths = 6
```