CSC 1351: Exam 2: Midterm

VERSION C

1 Recursion

1.1

From CodingBat.
Given a string, compute recursively a new string where all the adjacent chars are now separated by a "*".

```
allStar("hello") -> "h*e*l*l*o"
allStar("abc") -> "a*b*c"
allStar("ab") -> "a*b"
```

```
  public static String allStar(String str) {
    return allStar(0,str);
  }
  public static String allStar(int n,String str) {
    if(n == str.length()) {
      return "";
    } else if(n+1 == str.length()) {
      return ""+str.charAt(n);
    } else {
      return ""+str.charAt(n)+"*"+allStar(n+1,str);
    }
  }
```

1.2

Rewrite the following code to use recursion (no loops!).

```
public class Hello {
  public static void hello(int n) {
    for(int i=0;i<n;i++)
      System.out.println("Hello␣"+i);
  }
  public static void main(String[] args) {
    hello(3);
  }
}
```

```
public class Hello {
  public static void hello(int n) {
    hello(0,n);
  }
  public static void hello(int i,int n) {
    if(i<n) {
      System.out.println("Hello␣"+i);
      hello(i+1,n);
    }
  }
  public static void main(String[] args) {
    hello(3);
  }
}
```

1.3

From CodingBat:
Given a string, compute recursively a new string where all the lowercase 'x' chars have been moved to the end of the string.

```
endX("xxre") -> "rexx"
endX("xxhixx") -> "hixxxx"
endX("xhixhix") -> "hihixxx"
```

```
  public static String endX(String x) {
    return endX(0,x);
  }
  public static String endX(int n,String x) {
    if(n == x.length()) return "";
    if(x.charAt(n) == 'x')
      return ""+endX(n+1,x)+"x";
    return ""+x.charAt(n)+endX(n+1,x);
  }
```

1.4

This task involves a game with plush bunnies. You start the game with a given number of bunnies. You can then give back some of the bunnies, according to the rules below. If multiple rules apply, it is your choice which to use. If none applies, you loose. The goal is to end up with exactly 42 bunnies. n is the number of bunnies you currently have.

- If n is even, you may give back exactly n/2 bunnies.
- If n is divisible by 3 or 4, you may multiply the last two digits of n and give back this many bunnies. (the last digit of n is n%10, the next-to-last digit is ((n%100)/10).
- If n is divisible by 5, then you may give back exactly 42 bunnies.

For example, suppose you start with 250 bunnies, you could do the following moves:

- Since 250 is divisible by 5, you may return 42 bunnies, leaving you with 208.
- Since 208 is even, you may return half of them, leaving you with 104.
- Since 104 is even again, you may do the same again, leaving you with 52.
- Since 52 is didisible by 4, you may multiply the last two digits (2*5=10), and return 10 bunnies, leaving you with 42 bunnies, and resulting in a win.

Write a recursive function to meet this specification:

```
// The return value should indicate whether it is possible to win this game if
// you start with n bunnies. For example:
// bunnies(250) is true
// bunnies(42) is true
// bunnies(84) is true
// bunnies(53) is false
// bunnies(41) is false
public class Bunnies {
    public static boolean bunnies(int n) {
        // smaller than 42
        if (n<42) return false;
        // equal 42
        if (n==42) return true;
        // divisible by 2
        if (n%2==0 && bunnies(n/2)) return true;
        // divisible by 3 or 4
        if ((n%3==0||n%4==0) && bunnies(n-((n%10)*((n%100)/10)))) return true;
        // divisible by 5
        if (n%5==0 && bunnies(n-42)) return true;
        return false;
    }
```

2 Sorting

2.1

Fill in the missing code. The number of lines corresponds to the answer key. Your code may vary.

```
public class Starship implements Comparable {
  String name; // sort by this first
  int range_in_parsecs ; // sort by this if the names are the same
  @Override
  public int compareTo(Object o) {
    Starship that = (Starship)o;
    ..........
    int diff = this.name.compareTo(that.name);
    ..........
    if(diff != 0) return diff;
    ..........
    return this.range_in_parsecs - that.range_in_parsecs;
    ..........
  }
  public static void main(String[] args) {}
}
```

2.2

What kind of sort is this? Rewrite it to use ArrayList.

```
  static void sort(final int start,final int end,int[] values) {
    if(end-start < 1)
      return;
    int[] scratch = new int[end-start+1];
    int r = rand.nextInt(scratch.length)+start;
    int p = values[r];
    int lo = 0, hi = scratch.length;
    int numps = 0;
    for(int i=start;i<=end;i++) {
      if(values[i] < p)
        scratch[lo++] = values[i];
      else if(values[i] == p)
        numps++;
      else
        scratch[--hi] = values[i];
    }
    int n = start;
    for(int i=0;i<lo;i++)
      values[n++] = scratch[i];
    for(int i=0;i<numps;i++)
      values[n++] = p;
    for(int i=hi;i<scratch.length;i++)
      values[n++] = scratch[i];
    sort(start,start+lo-1,values);
    sort(start+hi,end,values);
  }
```

The random number generation gives it away. This is a quicksort.

```
  static void sort(final int start,final int end,ArrayList<Integer> values) {
    if(end-start < 1)
      return;
    int[] scratch = new int[end-start+1];
    int r = rand.nextInt(scratch.length)+start;
    int p = values.get(r);
```

```
    int lo = 0, hi = scratch.length;
    int numps = 0;
    for(int i=start;i<=end;i++) {
      if(values.get(i) < p)
        scratch[lo++] = values.get(i);
      else if(values.get(i) == p)
        numps++;
      else
        scratch[--hi] = values.get(i);
    }
    int n = start;
    for(int i=0;i<lo;i++)
      values.set(n++,scratch[i]);
    for(int i=0;i<numps;i++)
      values.set(n++,p);
    for(int i=hi;i<scratch.length;i++)
      values.set(n++,scratch[i]);
    sort(start,start+lo-1,values);
    sort(start+hi,end,values);
  }
```

2.3

Find the best match between the column on the left and the column on the right.

1 recursion	a can always be rewritten as recursive
2 binary search	b is $O(\log N)$
3 bubble sort	c is $O(N \log N)$
4 quick sort	d uses a random number
5 iterative	e keeps finding the next smallest element
6 merge sort	f compares adjacent elements
7 selection sort	g can cause StackOverflowError

3 Anonymous Inner Classes

3.1

Given the interface defined like this:

```
public interface Logger {
  void log(String msg);
}
```

Write a complete program that uses an anonymous inner class that implements this interface to print the message msg to the screen.

```
interface Logger {
  void log(String msg);
}
public class Test {
  public static void main(String[] args) {
    Logger log = new Logger() {
      public void log(String msg) {
        System.out.println(msg);
      }
    };
    log.log("Hello␣World");
  }
}
```

4 Big O

4.1

What is the Big-O notation for the following function?
$T(N) = 100000N^3 + 10N + N^{-2}$
The leading term as N gets large is N^3, so...
$O(N^3)$