COS 126 **General Computer Science** **Fall 2008**

Exam 1 Solutions

1. **Number representation**

   (a)
   ```
   2009:  0111 1101 1001
   2010:  0111 1101 1010
   2011:  0111 1101 1011
   2012:  0111 1101 1100
   ```

   (b) **20**

   (c) **20**

   (d) FFF5 (hex) = -11 (decimal)

   (e) 19 (decimal) = 13 or 0013 (hex)

2. **Short Answer**

   (a)
   ```
   % javac A.java
   % javac B.java
   % java A | java B
   ```

   (b) N^3

   (c) ii. Binary search of a sorted array of length N

   (d) `XORmystery` swaps the values of `a` and `b`

3. **Methods**

   (a)
   ```
   % java HIthere 10
   Bonjour, monde.
   ```

   (b)
   ```
   % java HIthere 7
   Hello, monde.
   ```

4. **Arrays and Loops**

```
public class CountingSort {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);      // Number of integers
*       int C = Integer.parseInt(    args[1]      );   // Maximum value

        // Allocate an array for each value in the range 0 to C
*       int[] counts = new int[   C+1       ];

        // Read the list of integers and count the number of times each
        // value appears.
        for (int i = 0; i < N; i++) {
*           int num = Integer.parseInt(       args[i+2]           );
*           counts[        num         ] += 1;
        }

        // Go through the counts array in order and print out the sorted list!
*       for (int i = 0;     i <= C         ; i++) {
            // Each time an integer appeared with this value, print it out.
            for (int j = 0; j < counts[i]; j++) {
*               System.out.print(   i     + " ");
            }
        }
        System.out.println();                   //to make the output pretty
}
```

5. **Input and Output**

```
public class Stats {

    public static void main(String[] args) {

        // read in the host to compare
        String hostName = args[0];
        // sum of all CPU usages
        double sum = 0;
        // counter for how many CPU uses for that host
        int counter = 0;

        while(!StdIn.isEmpty()) {
            // read in one sample
            String currentHost = StdIn.readString();
            double currentCPU = StdIn.readDouble();

            if(currentHost.equals(hostName)){
                // add to the CPU sum
                sum += currentCPU;
                counter++;
            }
        }

        // calculate the average CPU use
        double avgCPU = sum/counter;
        System.out.println("AvgCPU: " + avgCPU);
    }
}
```

6. **Debugging**

   (a) `7` `int temp = a[newpos];`

   (b) `3` `for (int i=0; i < a.length; i++);`

   (c) random re-arrangement (or shuffle) of elements of an array

   (d) `5` `int newpos = (int) (Math.random() * a.length);`

7. **Recursion**

   (a) They both compute x^N

   (b) 10

   (c) 5

   (d) 9

   (e) N

   (f) $logN$

8. **Recursive Graphics**

   (a) F

   (b) D

   (c) A

9. **Efficient Algorithms**

   (a) $f(3) = 2$

   (b)
```
*    //Conway Sequence,            recursive               version
     public class Conway1 {

       public static int con(int n) {
*        if (n <= 2) return 1;          //     base case
*        return con(con(n-1)) + con(         n-con(n-1)               );
       }

       public static void main(String[] args) {
         int n = Integer.parseInt(args[0]);
         System.out.println( con(n) );
       }
     }

*    //Conway Sequence,          dynamic programming         version
     public class Conway2 {

       public static void main(String[] args) {
         int n = Integer.parseInt(args[0]);
         // con array stores intermediate results
         int[] con = new int[n+2];  //oversized so it works on n = 1 or 2
         con[1] = 1;
         con[2] = 1;
         for (int i = 3; i <= n; i++) {
*          con[i] = con[con[i-1]] +             con[i-con[i-1]]        ;
         }
         System.out.println( con[n] );
       }
     }
```

   (c) Second is faster because it avoids repeated function calls.

10. **TOY programming**

   (a)
```
0001
0002
0004
0008
0010
0020
```

   (b) `4000`

   (c) Outputs in order all the powers of 2 it can represent.