COS 126	General Computer Science	Fall 2007

Exam 1

This test has 10 questions worth a total of 50 points. You have 120 minutes. The exam is closed book, except that you are allowed to use a one page cheatsheet, 8.5 by 11 inches, one side only, handwritten by you. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. Partial credit will be given for partially correct answers. **Write out and sign the Honor Code pledge before turning in the test:**

"I pledge my honor that I have not violated the Honor Code during this examination."

_____

Signature

Problem	Score
0	
1	
2	
3	
4	
Sub 1	

Problem	Score
5	
6	
7	
8	
9	
Sub 2	

Total	

Name:

NetID:

Preceptor: Donna Adam

Shirley JP

Tom Ed

Corey Zia

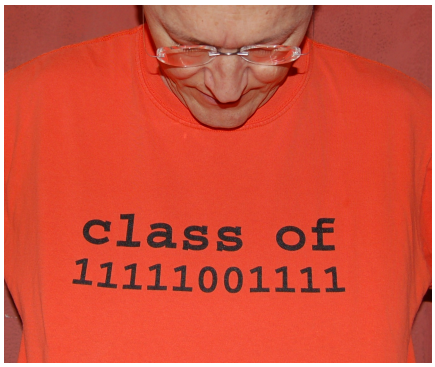
Martin

0. **Miscellaneous. (2 points) (really)**

   (a) Write your name and Princeton NetID in the space provided on the front of the exam, and circle the name of your preceptor.

   (b) *Write* and sign the honor code on the front of the exam.

1. **Number representation (4 points)**



   (a) Each year, Princeton Computer Science majors design a W T-shirt. To the left is the highly creative binary effort from a recent class. Which class? (in decimal, we mean) For 1 extra point, explain how to figure it out with just two or three subtractions and no additions.

   (b) An extra-geeky class might even have used hexadecimal instead of binary. What would the shirt read then?

   (c) In what year will you (or did you) graduate from college? Write it down in decimal and binary.

2. **Short Answer (3 points)**

(a) Many years ago programmers used to write the following code: `y = y ^ y;` believing it accomplished a certain operation particularly quickly. What operation is this code designed to accomplish? (Remember that `^` is the Java operator for exclusive-OR.)

(b) What will the following code print?

```
public static void changeGrade(int grade) {
   grade = grade - 10;
}

public static void main(String[] args) {
   int grade = 90;
   changeGrade(grade);
   System.out.println(grade);
}
```

(c) A newly-discovered comet is heading for Earth. Princeton astronomers say that a very small uncertainty in our knowledge of its location will cause a huge uncertainty about whether it will collide with Earth or not.

Accurate calculation of the comet's future trajectory:

i. is an ill-conditioned problem

ii. will take time proportional to N^3, where N is the number of planets, comets and asteroids in the orbit of the comet

iii. is an example of a numerically unstable algorithm

iv. will take time proportional to $NlogN$, where N is as in ii above, but the planets, comets, and asteroids have first been sorted in order of mass

Please circle your answer.

This page is intentionally left blank.

3. **Input and Output (8 points)**

A DNA sequence can be written as a string where the characters denote the following bases A = adenine, T= thymine, C = cytosine, and G = guanine. DNA has the interesting property that A always pairs with T and C always pairs with G. Often you are given just one strand of DNA, but you would like to work with the complementary strand, the strand that pairs with the single strand of DNA that you are working with. This problem asks you to write a program that prints the sequence of the complementary strand.

(a) Given the following string which represents a DNA sequence what would its complementary strand be?

`ATGCATGC`

(b) Write code that reads in a legal DNA sequence (such as in (a) of this question) from standard input, one character at a time, and outputs the complementary strand of DNA. (A `char` is a primitive data type that represents one character, so you may use the conditional operators `==` and `!=` as needed. A `char` constant must use single quotes, e.g., `char x = 'A';` The method `StdIn.readChar()` will read in one character at a time. The input characters will not be separated by white space.)

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









  }
}
```

(c) Write the command you would use to run this program reading input from the file `sequence.txt`.

4. **Arrays and Loops (8 points)**

Below is a partially complete program that is supposed to do the following: from standard input read an integer N followed by a sequence of N records of cars sold by a dealership in 2006. Each record consists of four fields, separated by whitespace: Make, Model, Dealer Cost, and Sale Price. The program then prints the record of the individual car sale that earned the highest profit. (This is New Jersey, so you may assume that no cars were sold at a loss.)

For example, if this were the input file `inventory.txt`:

```
5
Honda    Civic  10000.00 15000.00
Nissan   Maxima 20000.00 35000.00
Nissan   Altima 20000.00 25000.00
Ford     Focus 10000.00 12000.00
Chrysler Aspen 30000.00 32000.00
```

then the program should behave thusly:

```
%  java Inventory < inventory.txt
%  Make: Nissan, Model: Maxima, Cost: 20000.0, Sale: 35000.0, Profit: 15000.0
```

(The single decimal place in the output is correct. Don't try to format the dollar outputs.)

The program starts below and continues on the next page; just fill in the blanks to complete it. In case of a profit tie, by the way, just pick any car with the maximum profit.

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

        // read the number of records
        int N = StdIn.readInt();

        // initialize four parallel arrays
        String[] make     = new String[N];
        String[] model    = new String[N];
        double[] cost     = new double[N];
        double[] sale     = new double[N];

        double profit;
        int maxIndex = 0;               // index of item with max profit
        double maxProfit = 0.0;        // max profit
```

4. **Arrays and Loops (continued from previous page)**

```
        for (int i=0;_____________________________________) {

            // read in the data

            ________________________________________________

            ________________________________________________

            ________________________________________________

            ________________________________________________

            // calculate the profit

            profit = ________________________________________

            // save index of record with max profit

            if(_____________________________________________) {

              maxIndex =  ___________________________________

              maxProfit = ___________________________________
            }
        }

        // print the record of the item with the maximum profit

        System.out.println(_____________________________________

                           ___________________________________

                           ___________________________________

                           ___________________________________

                           ___________________________________);
    }
}
```

5. **Recursion (6 points)**

Consider the following program.

```
1:public class RecursionQuestion {
2:
3:    public static int mystery(String[] a, int x) {
4:        System.out.println(a[x]);
5:
6:        // Base case
7:        if( x < 2 ) { return x; }
8:
9:        // Recursive case
10:
11:        if( x % 2 == 1 ) {                          // x is odd
12:            return mystery(a, (x - 1) / 2) - 1;
13:        }
14:        else {                                      // x is even
15:            return mystery(a,  x / 2) + 1;
16:        }
17:    }
18:
19:    public static void main(String[] args) {
20:        String[] a = {"The",  "truth", "is", "rarely", "pure", "and",
21:                      "never", "simple"};
22:
23:        int param = 0;
24:
25:        int result = mystery(a, param);
26:        System.out.println(a[result]);
27:
28:    }
29:}
```

(a) What does the program print out?

5. **Recursion (continued from previous page)**

   (b) Suppose we change line 23 to

   ```
   int param = 5;
   ```

   Please fill in the following recursive call trace, given this change.

   ```
   mystery (a, 5)

           mystery (a, ____ )

                   mystery (a, ____ )

                   return 1

           return ____

   return ____
   ```

   (c) Using your answers from (b), what does the program print out? (Do not forget the final print statement from the main function!)

6. **Sorting and Analysis (6 points)**

(a) Consider the following code, which sorts the N elements of an array of `int`s:

```
public class CoolSorts {

    public static void bubbleSort(int[] a) {
    // Make repeated passes through array
    // swapping adjacent elements if they are out of order.
    // When nothing needs to be swapped, array is sorted.

        int N = a.length;
        boolean swapped = true;
        while (swapped) {
            swapped = false;
            for (int i = 0; i < N-1; i++)
                if (a[i] > a[i+1]) {
                    swap(a, i, i+1);
                    swapped = true;
                }
        }
    }

    public static void swap(int[] a, int i, int j) {
        int t = a[i]; a[i] = a[j]; a[j] = t;
    }
}
```

i. For the best case (already sorted) input, the order of growth of the running time of this program will be which of the following functions of N? Circle one:

$$logN \qquad N \qquad NlogN \qquad N^2 \qquad 2^N \qquad N!$$

ii. For the *worst* case input, same question. (Hint: start by finding an upper bound for the number of times the `while` loop is executed.) Circle one:

$$logN \qquad N \qquad NlogN \qquad N^2 \qquad 2^N \qquad N!$$

6. **Sorting and Analysis (continued from previous page)**

(b) Murphy has proposed and written another sorting algorithm for the `CoolSorts` class. It also sorts integer arrays of size N:

```
public static void murphySort(int[] a) {
    while (!isSorted(a)) {
        // Swap elements randomly until everything is in sorted order.
        for (int i = 0; i < a.length; i++) {
            int j = i + (int) (Math.random() * (a.length - i));

            //same swap method as before
            swap(a, i, j);
        }
    }
}

public static boolean isSorted(int[] a) {
// return true if array elements are in sorted order
    for (int i = 1; i < a.length; i++) {
        if (a[i-1] > a[i]) return false;
    }
    return true;
}
```

i. Suppose Murphy is lucky and the input array is already sorted. In this best case scenario, the order of growth of the running time of this program will be which of the following functions of N? Circle one:

$$logN \qquad N \qquad NlogN \qquad N^2 \qquad 2^N \qquad N!$$

ii. Now suppose the input array is in a completely unsorted order, but Murphy is lucky in a different way: the program somehow never permutes the array into the same configuration twice. In this scenario, which of the following functions of N expresses the worst case order of growth of the running time of this program? Circle one:

$$logN \qquad N \qquad NlogN \qquad N^2 \qquad 2^N \qquad N!$$

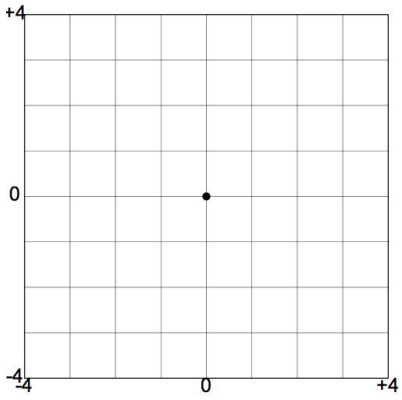
7. **Graphics, Functions, and Loops (6 points)**

Consider the following program, which uses Fibonacci numbers to draw a picture. You may assume that the `fib` function has been implemented in some clever way, but note that this one returns 0 if `n <= 0`. The function `StdDraw.square(x, y, radius)` draws a square centered at `(x,y)` with sides of length `2*radius`.

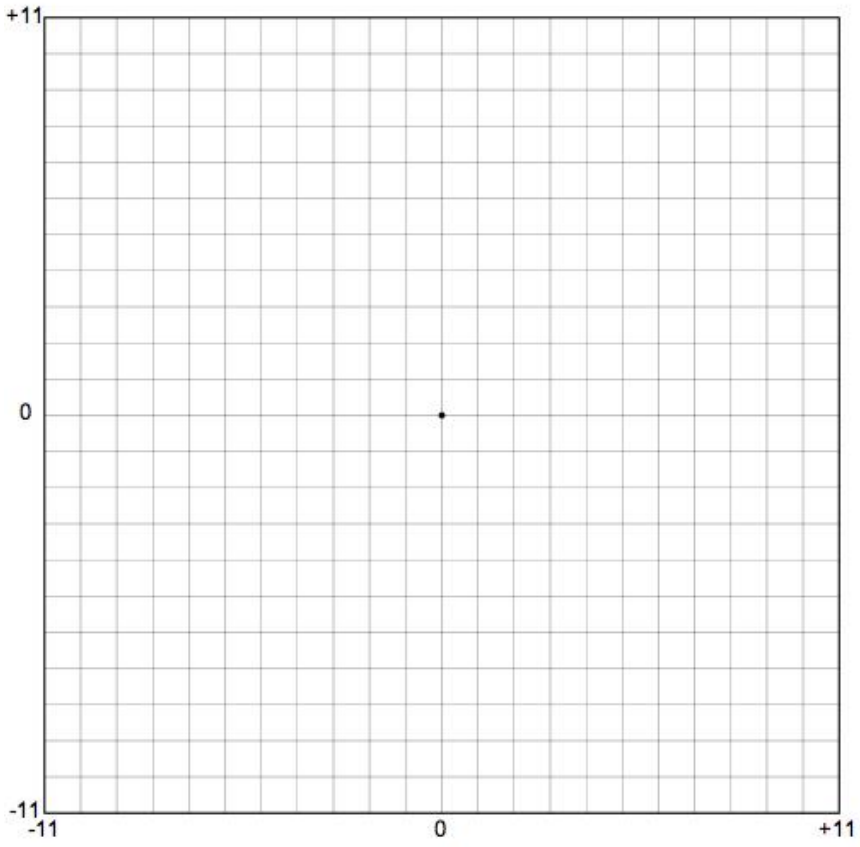
```
public class Shape {

    public static int fib(int n) {
        // Returns nth Fibonacci number for n > 0
        // fib(1) is 1; fib(2) is 1; fib(n) is sum of previous two
        // Returns 0 for n <= 0
    }

    public static void drawShape(int n) {
        double x = 0; double y = 0;

        for (int i = n; i > 0; i--) {
                double radius = fib(i);
                double small  = fib(i-2);
                double big    = fib(i+1);

                StdDraw.square(x, y, radius);
                int direction = i % 4;
                if (direction == 0) {            // go down
                     x += small;  y -= big;
                } else if (direction == 1) {     // go right
                     x += big;    y += small;
                } else if (direction == 2) {     // go up
                     x -= small;  y += big;
                } else if (direction == 3) {     // go left
                     x -= big;    y -= small;
                }
            }
        }

    public static void main(String[] args) {

        int n   = Integer.parseInt(args[0]);
        int rad = fib(n) + 2*fib(n-1);
        StdDraw.setXscale(-rad, rad);
        StdDraw.setYscale(-rad, rad);
        drawShape(n);
    }
}
```

7. **Graphics, Functions, and Loops (continued from previous page)**

   (a) What would appear if you ran the command: `java Shape 3` ? (You could try `java Shape 1` as a warmup.) Here's a handy grid you can write on:



   (b) What would appear if you ran the command: java Shape 5 ? And again a handy grid:



8. **Java Programming, Recursion, Functions, and Loops (5 points)**

   Here is Euclid's Algorithm, expressed as a recursive function, just as you saw in lecture. Please rewrite it as a nonrecursive function that uses a `while` loop.

```
public static int gcd(int p, int q) {
    if (q == 0) return p;
    else return gcd(q, p % q);
}
```

9. **Definition (2 points)**

   In *one* sentence, what do you think the word "algorithm" means? (2 points for any serious answer)