COS 126 **General Computer Science** **Fall 2006**

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, 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:

Nadia	Ari
George	Adam
Maia	Elliott
Tim	Vivek

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 the preceptor who grades your homework assignments.

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

1. **Short Answer (5 points)**

   (a) Here is a 16-bit two's complement binary integer: 1111111111101100. Convert it to decimal. Circle your answer.

   (b) Write the value of `(double) ( 22 / 7 )`.

   (c) Write the value of b after the following two statements are executed. Remember that Java ints use 32-bit 2's-complement representation:

   ```
   int a = 2147483647;    // 2^31 - 1
   int b = a + 1;
   ```

   (d) What is the result of the Java expression (17.0/0.0)?

   (e) Write this number using Java's scientific notation, (without using `Math.pow`):
   $6.022 \cdot 10^{23}$

2. **Arrays, Conditionals, Loops, and Bugs (5 points)**

   The following program is supposed to implement Bubblesort, a sorting algorithm that works by making repeated passes over an array, exchanging adjacent elements if they are out of order until in some pass it finds no elements out of order. The program takes a command-line parameter specifying how many random numbers should be generated. It then sorts the numbers, and prints them, showing how many passes were made over the array. The program is full of bugs. Find 5 for full credit. Circle each one and give a one sentence explanation beside each one.

```
public class Bubble {
    public static void MainProgram(String[] args) {
        int N = Integer.parseInt(args[0]);

        /* first allocate a bunch of numbers */
        double[] a;
        for (int i = 0; i < N; i++)
            a[i] = Math.random();

        boolean keepOn = false;
        int iters = 0;

        /* keep walking through array until sorted */
        while (keepOn) {
            keepOn = false;
            for (int i = 0; i < N; i++) {
                if (a[i] > a[i+1]) {
                    /* we found two unsorted elements - swap */
                    int temp = a[i];
                    a[i+1] = temp;
                    a[i] = a[i+1];
                    keepOn = true;
                }
            }
            iters++
        }

        for (int i = 0; i < N; i++)
            System.out.println(a[i]);

        System.out.println(iters + " iterations");
    }
}
```

3. **Functions (4 points)**

   Consider the following program

```
public class Functions {

    public static void printValues(int v) {
        System.out.println("This question is worth " + v + " points.");
    }

    public static void printValues(double v) {
        System.out.println("My score will be " + v + " points.");
    }

    public static void printValues(int x, double y) {
        System.out.print("I will get "+ x +" of ");
        System.out.println(y + " points on this question.");
    }

    public static void printValues(double x, int y) {
        System.out.print("Of all the " + x + " points, ");
        System.out.println("I will receive " + y + " of them.");
    }

    public static void main(String[] args) {

        // Initialize the variables
        int i = 10;
        double d = 9.5;

        // call some printing functions
        printValues(d);
        printValues(i);
        printValues(d, i);


    }
}
```

   What does it print out?

4. **Loops and Strings and Conditionals (6 points)**

Consider the following program.

```
public class Pal {
   public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        String r = "+";
        for (int i = 1; i <= N; i++) {
             if ((i % 2) == 0)
                 r = r + i + r;
             else r = i + r + i;
        }
        int len = r.length();
        System.out.println(r);
        for (int i = 1; i < len; i+=N) {
            for (int j = i; j < len; j+=2) {
                System.out.print(r.charAt(j));
                System.out.print("/");
            }
            System.out.println();
        }
        System.out.println(r.charAt(len));
   }
}
```

(a) What is the result of:

```
java Pal 3
```

(b) What is the order of growth of the running time of the program as a function of N? Circle one:

$logN$ N $NlogN$ N^2 2^N

5. **I/O (6 points)**

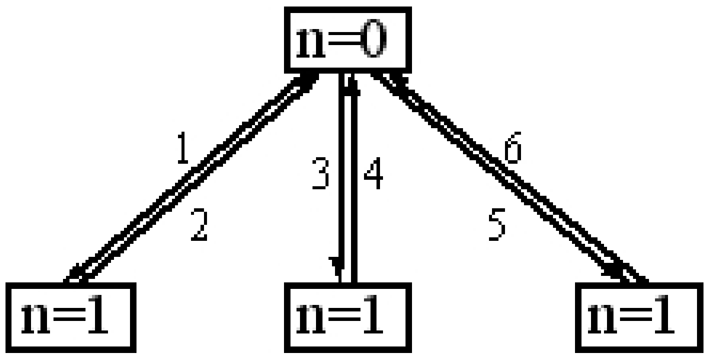
   (a) Write a function that takes a single argument X (a double), and then reads a sequence of doubles from standard input and prints out the one numerically closest to X. You do not need to use an array.

   (b) What is the order of growth of the running time of this function in terms of N, the number of doubles read in? Circle one:

   $logN$ N $NlogN$ N^2 2^N

6. **Recursion (5 points)**

   Given the Sierpinski recursive algorithm called to a depth of 1, the tree of the recursive calls and the order in which they happen is given below.



   Using the same method above, numbering the calls and returns in order, as well as the value of n, provide the tree for the recursive function S below:

```
public static void S(int n, int max) {

   if (n >= max) return;

   S(n + 1, max);
   S(n + 2, max);
   S(n + 3, max);
}
```

   for the function call `S(0, 3)`

7. **Java Programming and Graphics (6 points)**

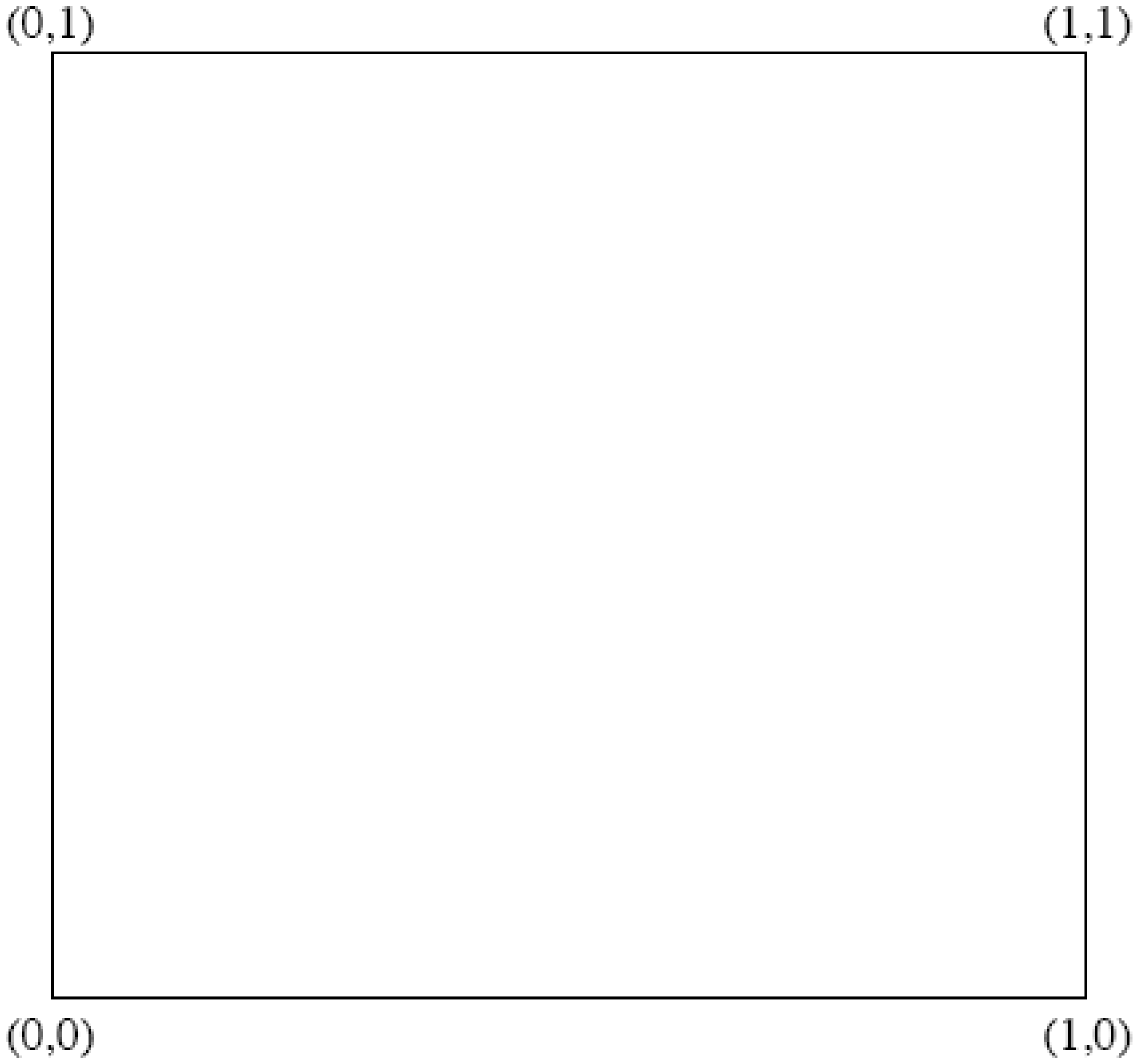
You have been using the StdDraw library for creating graphics images. Consider the following program that uses StdDraw.

```
public class MysteryPicture {

   public static void main(String[] args) {

      int N = Integer.parseInt(args[0]);
      double r = 1.0 / ((double)N * 2.0);
      double startX = 0.5;
      double startY = r;
      for (int i = 1; i <= N; i++)
      {
         double currX = startX;
         for (int j = 1; j <= i; j++)
         {
            if (j % 2 == 0)
            {
               StdDraw.circle(currX, startY, r);
            }
            else
            {
               StdDraw.filledCircle(currX, startY, r);
            }
            currX += r * 2.0;
         }
         startX -= r;
         startY += r * 2.0;
      }
   }
}
```

Continued on the next page . . .

7. **Java Programming and Graphics (continued from previous page)** What picture is produced by running the following command? Sketch it in the box below.

```
% java MysteryPicture 4
```



8. **Recursion (6 points)**

Consider the following recursive function:

```
public static int Ack(int m, int n) {
  if (m == 0) {
     return n + 1;
  }
  else if (n == 0) {
     return Ack(m - 1, 1);
  }
  else {
     return Ack(m - 1, Ack(m, n - 1));
  }
}
```

(a) What is the return value when the above function is run with

i. `Ack(0, 0)`

ii. `Ack(0, 3)`

iii. `Ack(0, 6)`

iv. What function of n does Ack(0, n) compute? (e.g., n * n + 7, 12n/5, etc.)

(b) What is the return value when the above function is run with

i. `Ack(1, 0)`

ii. `Ack(1, 3)`

iii. `Ack(1, 6)`

iv. What function of n does Ack(1, n) compute?

Continued on the next page . . .

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

(c) What is the return value when the above function is run with

i. `Ack(2, 0)`

ii. `Ack(2, 3)`

iii. `Ack(2, 6)`

iv. What function of n does Ack(2, n) compute?

9. **TOY Programming (5 points)**

   There are many instances of single TOY instructions that do nothing: so-called “no-ops” that change no register or memory location or standard output. You saw one example in lecture. For full credit, show how seven different opcodes can be used to make a TOY no-op instruction. Just write down one hex instruction for each opcode you choose, and feel free to include the one from lecture. (The TOY Reference Card is on the next page.)

```
             TOY REFERENCE CARD


INSTRUCTION FORMATS

            | . . . . | . . . . | . . . . | . . . .|
  Format 1: | opcode  |    d    |    s    |    t    |  (0-6, A-B)
  Format 2: | opcode  |    d    |       addr        |  (7-9, C-F)


ARITHMETIC and LOGICAL operations
    1: add              R[d] <- R[s] +  R[t]
    2: subtract         R[d] <- R[s] -  R[t]
    3: and              R[d] <- R[s] &  R[t]
    4: xor              R[d] <- R[s] ^  R[t]
    5: shift left       R[d] <- R[s] << R[t]
    6: shift right      R[d] <- R[s] >> R[t]

TRANSFER between registers and memory
    7: load address     R[d] <- addr
    8: load             R[d] <- mem[addr]
    9: store            mem[addr] <- R[d]
    A: load indirect    R[d] <- mem[R[t]]
    B: store indirect   mem[R[t]] <- R[d]

CONTROL
    0: halt             halt
    C: branch zero      if (R[d] == 0) pc <- addr
    D: branch positive  if (R[d] >  0) pc <- addr
    E: jump register    pc <- R[d]
    F: jump and link    R[d] <- pc; pc <- addr


Register 0 always reads 0.
Loads from mem[FF] come from stdin.
Stores to mem[FF] go to stdout.
```