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

Exam 1

This test has 11 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. 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	
10	
Sub 2	

Total	

Name:

NetID:

Preceptor:

Ana	David
Donna	Jeff
Maia	Mike
Sid	Tao
Tom	Will
Wyatt	

The TOY reference card is on the last page of the exam.
Feel free to tear out the last page in order to use it more easily.

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 (5 points)**

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

   (b) How many **zeroes** are in the binary representation of 2^{20} ?

   (c) How many **ones** are in the binary representation of $2^{20} - 1$?

   (d) Here's a (hex) integer in TOY's 16-bit, 2's-complement representation: FFF5
   What is it in **decimal**?

   (e) Please convert the decimal integer 19 into TOY's 16-bit, 2's-complement representation. Use **hex**, not binary.

2. **Short Answer (4 points)**

(a) A program `A.java` is designed to produce outputs that can be read in by another program, `B.java`. Write below the three command lines you would use to compile and execute `A` and `B` so that the output of `A` becomes the input of `B`, without using an intermediate data file:

```
%
%
%
```

(b) Your program has the following running times for different sizes of its input N:

```
N        Time (seconds)
10000         3
20000        24
30000        81
40000       192
```

What is the order of growth of the running time of this algorithm? Circle one of the following:

$logN$ N $NlogN$ N^2 N^3 2^N

(c) Which of the following algorithms has an order of growth of $logN$? Circle your answer(s).

i. N-body simulation
ii. Binary search of a sorted array of length N
iii. Matrix multiplication of two NxN arrays
iv. Reversing an array of length N
v. None of the above

(d) Consider the following program, which uses the Java bit-wise exclusive or operator ^:

```
public class XORmystery {
    public static void main(String[] args) {
        int a = StdIn.readInt();
        int b = StdIn.readInt();
        a = a ^ b;
        b = b ^ a;
        a = a ^ b;
        System.out.println(a);
        System.out.println(b);
    }
}
```

What does this program do?

3. **Methods (2 points)**

   Consider the following Java program

```
public class HIthere {

  public static void greeting(int x)
  {
    if (x < 10)
    {
       System.out.print("Hello,");
    }
    else
    {
       System.out.print("Bonjour,");     // French for "Hello"
    }
    x = x - 5;
  }

  public static void main(String[] args)
  {
    int x = Integer.parseInt(args[0]);
    greeting(x);
    if (x < 3)
    {
      System.out.println(" world.");     // Anglais pour "monde"
    }
    else
    {
      System.out.println(" monde.");
    }
  }
}
```

   What will be printed by this program in the following cases?

   (a) `% java HIthere 10`

   (b) `% java HIthere 7`

4. **Arrays and Loops (6 points)**

In this problem, you will fill in missing parts of an implementation of the *CountingSort* algorithm. CountingSort is used to sort N integers whose values are guaranteed to be between 0 and C, inclusive (i.e., as small as 0 and as large as C). It does this by counting the number of times each value appears in the list of integers, and then printing out these integers in order of increasing value. The CountingSort program below expects the following input from the command line: the number of integers in the list, the maximum value of any integer, and the list of integers. For example, the command:

```
% java CountingSort 5 10 9 2 10 6 6
```

should produce the following (sorted) output:

```
2 6 6 9 10
```

Fill in the **six** blanks in the program below. Use the comments as a guide for your answers. Lines with blanks to fill in are marked with an asterisk.

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

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

        // 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(                  );
*           counts[                 ] += 1;
        }

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

5. **Input and Output (6 points)**

   Write a java program `Stats.java` that reads in a series of sample measurements from standard input and prints out the average CPU usage of a specific hostname given as a command line argument.

   Each line of the input file represents one sample, and consists of the host name (a `String`) and that host's CPU usage for that sample (a `double`). Here is an example of a short input file `data.txt`:

```
opus.cs.princeton.edu 10.3
tux.cs.princeton.edu  15.9
opus.cs.princeton.edu  1.5
fred.cs.princeton.edu  9.0
```

   Here is the way your program should behave:

```
% java Stats opus.cs.princeton.edu < data.txt
  AvgCPU: 5.9
%
```

   Remember that to test whether two `String`s `x` and `y` are equal, you must say `(x.equals(y))` and not `(x == y)`. You will probably want to use these `StdIn` methods: `StdIn.isEmpty()`, `StdIn.readDouble()`, and `StdIn.readString()`.

   Use the opposite page for your program.

Write your `Stats.java` in the space below:

6. **Debugging (4 points)** Consider the following buggy program:

```
1 public class Homer {
2     public static void bart(int[] a) {
3         for (int i = 0; i <= a.length; i++) {
4             //compute new position
5             int newpos = (int)Math.random() * a.length;
6             //swap entries
7             temp = a[newpos];
8             a[newpos] = a[i];
9             a[i] = temp;
10        }
11    }
12    // main goes here (see below)
   }
```

(a) You try to compile the program but this happens:

```
% javac Homer.java
Homer.java:7: cannot find symbol
symbol  : variable temp
location: class Homer
             temp = a[newpos];
             ^
```

Write the correct version of this statement below:

(b) The program now compiles, so you decide to test it by writing the following main program inside the Homer class:

```
12     public static void main(String[] args){
13         int [] x = { 1, 2, 3, 4, 5 };           //short test array
14         bart(x);
15         for (int j=0; j < x.length; j++) System.out.println(x[j]);
16     }
```

But when you run the program this happens:

```
% java Homer
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
    at Homer.bart(Homer.java:8)
    at Homer.main(Homer.java:14)
```

Please fix this bug by changing one statement in `bart`. Write the new statement below, including the line number:

(c) What is the `bart` function attempting to do?

(d) With the `ArrayIndexOutOfBoundsException` now taken care of, you're disappointed that every time you run your program you get the same result:

```
% java Homer
5
1
2
3
4
% java Homer
5
1
2
3
4
  and so on ...
```

Please fix this bug by changing one statement in `bart`. Write the new statement below, including the line number:

7. **Recursion (6 points)**

   Consider the following functions:

```
public class Mystery {

  public static double square(double x)
  {
    return(x * x);
  }

  public static boolean isEven(int x)
  {
    return((x % 2) == 0);
  }

  public static double mysteryOne(double x, int N)
  {
    if (N == 0)
    {
      return(1);
    }
    else
    {
      return(x * mysteryOne(x, N-1));
    }
  }

  public static double mysteryTwo(double x, int N)
  {
    if (N == 0)
    {
      return(1);
    }
    else if (isEven(N))
    {
      return(square(mysteryTwo(x, N/2)));
    }
    else
    {
      return(x * mysteryTwo(x, N-1));
    }
  }
}
```

(a) `mysteryOne` and `mysteryTwo` perform the same function. What is it?

(b) How many multiplications are needed to compute `mysteryOne(2.0, 10)` ?

(c) How many multiplications are needed to compute `mysteryTwo(2.0, 10)` ?

(d) How many multiplications are needed to compute `mysteryTwo(2.0, 100)` ?

(e) What is the order of growth of the running time of `mysteryOne()` ? Circle your answer:

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

(f) What is the order of growth of the running time of `mysteryTwo()` ? Circle your answer:

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

8. **Recursive Graphics (6 points)**

   The following program uses `StdDraw.filledSquare(x, y, r)`, which will draw a solid square of side length `2*r` centered on the `(x,y)` point.

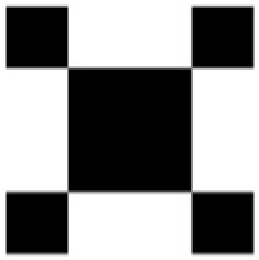
```
public class SquareArt {
    public static void fieldOfScreens(double x, double y, double radius,
                                      boolean fill, int level)
    {
        if (level == 0) {
            if (fill) StdDraw.filledSquare(x, y, radius);
            return;
        }
        else {
            double r = 0.5 * radius;
            fieldOfScreens(x - r, y - r, r, fill, level-1);
            fieldOfScreens(x - r, y + r, r, !fill, level-1);
            fieldOfScreens(x + r, y - r, r, !fill, level-1);
            fieldOfScreens(x + r, y + r, r, fill, level-1);
         }
    }
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        fieldOfScreens(0.5, 0.5, 0.5, true, N);
    }
}
```

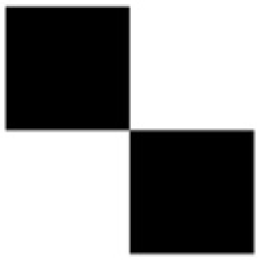
   From the choices on the opposite page, please select one for each of the following questions:

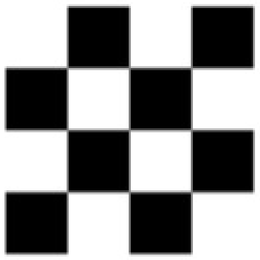
   (a) Which picture will be drawn by `java SquareArt 0`? Write the letter below:

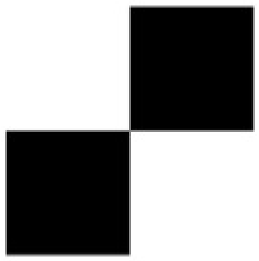
   (b) Which picture will be drawn by `java SquareArt 1`? Write the letter below:

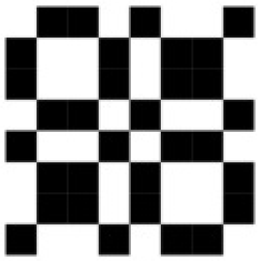
   (c) Which picture will be drawn by `java SquareArt 2`? Write the letter below:

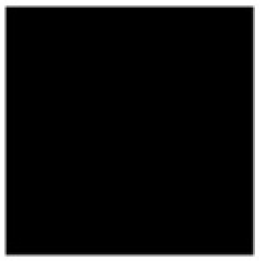
A: 

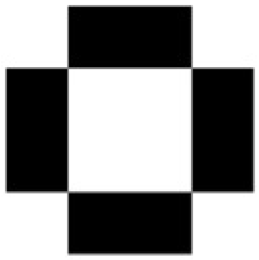
B: 

C: 

D: 

E: 

F: 

G: 

H: none of these pictures

9. **Efficient Algorithms (4 points)**

   The *Conway sequence* is defined thus:

   $f(n) = f(f(n-1)) + f(n - f(n-1))$ for $n > 2$ and $f(2) = f(1) = 1$.

   (a) Compute f(3)

   (b) Below are two incomplete java programs that compute the nth Conway number. Fill in the **five** blanks with appropriate comments or code. Lines with blanks to fill in are marked with asterisks.

```
*    //Conway Sequence,                           version
     public class Conway1 {

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

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

*    //Conway Sequence,                               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]] +                                    ;
         }
         System.out.println( con[n] );
       }
     }
```

   (c) Which one will run faster? Why?

10. **TOY programming (5 points)**

Suppose the following is the contents of TOY memory, and the PC is started at address 10.

```
00: 0001   1

10: 8A00   R[A] <- mem[00]
11: 9AFF   write R[A]
12: 1AAA   R[A] <- R[A] + R[A]
13: DA11   if (R[A] > 0) pc <- 11
14: 0000   halt
```

(a) What are the first 6 outputs, in hexadecimal, of this program?

(b) What is the last output, in hexadecimal, of this program? (Remember that TOY uses those 16-bit 2's complement integers.)

(c) What does this program do?

`This page is supposed to be blank.`

```
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.
pc starts at 10

16-bit registers
16-bit memory locations
8-bit program counter
```