

CSCI 136 Written Exam #1
Fundamentals of Computer Science II
Spring 2012

Name: _____

This exam consists of 5 problems on the following 8 pages.

You may use your double-sided hand-written 8 ½ x 11 note sheet during the exam. No computers, mobile devices, cell phones, or other communication devices of any kind are permitted.

If you have a question, raise your hand and I will stop by. Since partial credit is possible, **please write legibly and show your work.**

| Problem | Points | Score |
|----------------|---------------|--------------|
| 1 | 8 | |
| 2 | 12 | |
| 3 | 16 | |
| 4 | 12 | |
| 5 | 14 | |
| Total | 62 | |

1. **Loops, Conditionals, I/O** (8 points). Consider the following program:

```
public class Prob1
{
    public static void main(String [] args)
    {
        double d = Double.parseDouble(args[0]);
        int a = 0;
        for (int i = 1; i < args.length; i++)
        {
            if (Double.parseDouble(args[i]) > d)
                a++;
        }
        System.out.printf("%d out of %d\n", a, args.length - 1);
    }
}
```

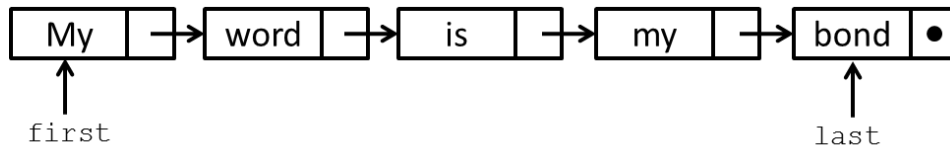
Below are four example executions of the program. Give the output produced by the program. If the given input would cause a runtime error, write "runtime error". If the given input would cause a stack overflow error, write "stack overflow".

| Command line | Output |
|------------------------------|---------------|
| % java Prob1 0.5 2.3 0.3 0.6 | 2 out of 3 |
| % java Prob1 0.5 | 0 out of 0 |
| % java Prob1 | Runtime error |
| % java Prob1 0.5 0 -1 2.0 3 | 2 out of 4 |

2. **Linked Structures** (12 points). Assume you have a linked list that holds Strings. It uses the following inner class `Node`:

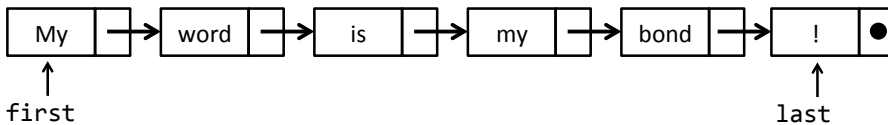
```
private class Node
{
    String item;
    Node next;
    Node(String s, Node n) { item = s; next = n; }
};
```

Currently the linked list has the following data and structure:

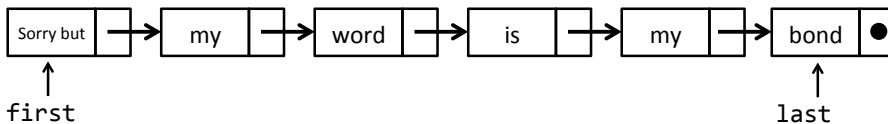


Draw the linked list resulting from running each of the following code segments. Be sure to **show where the *first* and *last*** variables are pointing. **Each part is independent** of the other parts (assume each part starts with the linked list shown in the above diagram).

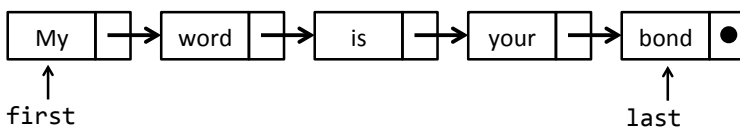
```
last.next = new Node("!", null);
last = last.next;
```



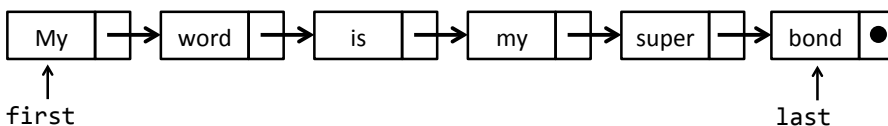
```
first = new Node("Sorry but", first);
```



```
for (Node c = first; c != null; c = c.next)
{
    if (c.item.equals("my"))
        c.item = "your";
}
```



```
for (Node c = first; c != null; c = c.next)
{
    if (c.item.equals("my"))
        c.next = new Node("super", c.next);
}
```



3. **Multiple choice** (2 points each). For each question, circle the **ONE** correct answer.

a) All of the following lines of code result in the variable `i` being increased by one on the line following the given line **EXCEPT**:

- I. `i++;`
- II. `++i;`
- III. `foo(i + 1);`
- IV. `i += 1;`
- V. `i = i + 1;`
- VI. `foo(i++);`

`foo` is a method with the signature `void foo(int val)`

b) Assume `vals` is a single dimension array that has been declared and instantiated to contain one or more elements of type `double`. Which of the following lines has the **potential** to generate an `ArrayIndexOutOfBoundsException` runtime exception?

- I. `vals[(int) Math.random() * vals.length] = 0.0;`
- II. `vals[vals.length - 1] = vals[0];`
- III. `vals[(int) (Math.random() * (vals.length - 1))] = 0.0;`
- IV. `vals[vals.length - 1] = vals[vals.length - 2];`

c) You are benchmarking an algorithm that takes an input of size `N`. For an input of size `N=50,000`, the algorithm took 4.1 hours. For an input of size `N=100,000` it took 16.2 hours. Which of the following best describes the order of growth of the algorithm:

- I. $O(1)$
- II. $O(N)$
- III. $O(N^2)$
- IV. $O(N^3)$
- V. $O(2^N)$

d) In Java socket programming, which of the following lines causes a **server** program to block (i.e. suspend its execution) until a client requests a new connection:

- I. `Socket sock = new Socket("localhost", 5000);`
- II. `Socket sock2 = sock.accept();`
- III. `InputStreamReader stream = new InputStreamReader(sock.getInputStream());`
- IV. `BufferedReader reader = new BufferedReader(stream);`
- V. `PrintWriter writer = new PrintWriter(sock.getOutputStream());`

e) You have a single instance of a Java class `Foo` that has three methods `m1`, `m2` and `m3`. Methods `m1` and `m2` are marked as `synchronized` while method `m3` is not. The program has several active threads. If method `m1` is currently being executed by one thread, which of the following is true:

- I. No other thread can be inside method `m1`, but one could be inside method `m2` or `m3`.
- II. No other thread can be inside method `m1` or `m2`, but one could be inside method `m3`.
- III. No other thread can be inside any method in class `Foo`.
- IV. A thread's `run()` method can call `join()` to wait for all other threads to exit `m1` and `m2`.

f) All the following are **reliable** conditions testing if the `String` variable `password` is "secret" **EXCEPT**:

- I. `(password.equals("secret"))`
- II. `(password.compareTo("secret") == 0)`
- III. `(password.matches("secret"))`
- IV. `(password == "secret")`
- V. `(new String("secret").equals(password))`

g) Which of the following is **true** about threading in Java:

- I. When a thread calls `Thread.sleep(1000)`, it will enter the running state and start executing on the CPU in exactly one second.
- II. You must have a multi-core CPU to run a multi-threaded Java program.
- III. The `Runnable` interface requires you implement a single method, namely `void run()`.
- IV. Catching an `InterruptedException` is optional when calling `join()` or `sleep()`.
- V. The `join()` method is used to copy output data from a worker thread.

h) You have a large array of N integers. You want to know if a given value is in that array. Which of the following is **true** about using a recursive binary search algorithm to look for the value:

- I. $\log_2(N)$ worker threads are required to perform the search in $O(\log N)$ time.
- II. The array must not have any repeated values.
- III. The array must be sorted.
- IV. In the worst case, binary search may need to make N^2 passes over the array elements.
- V. All the array values must be nonnegative.

4. **Recursion** (12 points). Consider the following recursive method:

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

What value is returned for the call `mystery(7, 3)`?

21

What function does `mystery()` compute for positive `n` and `m`?

`n` multiplied by `m`

5. **Threads** (14 points). Consider the following multi-threaded program:

| | |
|---|---|
| <pre>public class Nums { public static void main(String [] args) { int N = Integer.parseInt(args[0]); int M = Integer.parseInt(args[1]); double [] a = new double[N]; for (int i = 0; i < N; i++) a[i] = StdIn.readDouble(); Thread [] threads = new Thread[M]; for (int i = 0; i < M; i++) { Worker w = new Worker(a, i, M); threads[i] = new Thread(w); threads[i].start(); } try { for (Thread t : threads) t.join(); } catch (InterruptedException e) { e.printStackTrace(); } for (double d: a) System.out.printf("%.2f ", d); } }</pre> | <pre>public class Worker implements Runnable { private double [] d; private int start; private int stride; public Worker(double [] d, int s, int t) { this.d = d; this.start = s; this.stride = t; } public void run() { for (int i = start; i < d.length; i = i + stride) { d[i] = Math.sqrt(d[i]); } } }</pre> |
|---|---|

a) What does N control in the `Nums` program?

N is the number of double values that are read into the array from standard input.

b) What does M control in the `Nums` program?

M controls the number of worker threads that are created.

c) What is the goal of the program?

Compute the square root of each of N double value read in from standard input. It does this in parallel using M worker threads.

(continued on next page)

5. **Threads** (continued)

d) What is a possible advantage of using $M > 1$ in this program?

On a multi-core/multi-processor computer, the threads can all work in parallel on taking the square root of different elements in the array. This could potentially speed up the calculation.

e) Assume a user executes “`java Nums 5 2`” and then enters the numbers 1, 4, 9, 16, and 100. What is the console output of the program?

1.00 2.00 3.00 4.00 10.00

f) Assume you removed the for-loop containing the `t.join()` line from `Nums`. What problem might this cause?

The output loop might run before all the worker threads have finished taking the square root.