Spring 2013 COMP 110-003 Midterm Exam Solutions

March 07, 2013

UNC Honor Pledge: I certify that no unauthorized assistance has been received or given in the completion of this work.

Signature:_____

Read this entire first page before beginning the exam.

You may NOT use any course materials in completing this exam.

You will have 75 minutes to complete this exam.

The exam consists of three sections: first, a section of multiple choice questions and short answer questions; second, a section requiring you to trace a short piece of code and write the output produced by the code; finally, a section requiring you to write code.

1) The second section provides three questions of different points. You may choose any TWO questions from them. If you answer all three questions, the two on which you receive the highest scores will be counted in your final score.

2) The third section provides three questions of different points. You may choose any TWO questions from them. If you answer all three questions, the two on which you receive the highest scores will be counted in your final score.

3) If your final score is greater than 100, the extra points will be counted.

Pace yourself! If you find that you are stuck for a long time on a question that is not worth many points, move on to another question that you know how to do.

Please write your name on every page of the exam. If you use the back side of the paper, write down clearly what question you are answering. Also, make sure to indicate the location of your answer under the question.

If something is unclear, raise your hand and ask.

Write clearly! Illegible answers will not receive credit.

Part 1. (40 points) Answer ALL questions.

1. (1 points) What are the two possible values of a bit?

   Answer: 0 and 1.

2. (1.5 points) What is the smallest addressable unit of memory? How many bits are in this unit?

   Answer: Byte. 8 bits.

3. (3 points) What does a compiler do? Why do we need a compiler?

   Answer: A compiler translates the code written in a programming language to machine instructions. We need it because it's hard for human beings to read and write machine instructions in bits directly.

4. (2.5 points) List 5 primitive types.

   Answer: Select any five from `int, byte, short, long, float, double, char, boolean.`

5. (3 points) Which of the following are legal variable names in Java (circle them)?

   ```
   bestFriend      7daysAWeek      hello!        TOTAL&COST
   private         FIFTYSEVEN57    do            new
   ```

   Answer: `bestFriend` and `FIFTYSEVEN57`.

6. (5 points) What are the values of the following variables?

   (a) ____3.0____ `double var1 = 10 / 3;`

   (b) ____6____ `int var2 = (int) (2.5 * 2.6);`

   (c) ____true____ `boolean var3 = !(3 > 3);`

   (d) ____true____ `boolean var4 = (121 % 11 == 0) || (5 == 5);`

   (e) ____2____ `int var5 = 11 % 3;`

7. (3 points) Declare a variable to hold the amount of money in your bank account and initialize it to 245.25. Name your Java variable in a meaningful way so that any programmer would know what value it holds. Your variable name should be at least two words.

   Answer: `double accountBalance = 245.25;`

8. (3 points) Assuming that `a` and `b` are both `int` type variables. Describe the difference between "`a = b`", "`a == b`" and "`a += b`".

   Answer: "`a = b`" assigns the value of `b` to the variable `a`; "`a == b`" is a boolean expression and its value represents whether the value of `a` is equal to the value `b`; "`a += b`" is also an assignment statement which is in fact "`a = a + b`".

9. (2 points) Suppose you have the variable `str` of type String with data "How are you?" Write out the data stored in `str` and place the index of each character below the string.

Answer:

H	o	w		a	r	e		y	o	u	?
0	1	2	3	4	5	6	7	8	9	10	11

10. (6 points) Give the **return type** and **value** of the following method calls on `str` (defined in the above question).

    (a) `str.length()`
    return type: int, value: 12

    (b) `str.equalsIgnoreCase("HOW ARE YOU")`
    return type: boolean, value: false

    (c) `str.indexOf("ou")`
    return type: int, value: 9

    (d) `str.lastIndexOf(" ")`
    return type: int, value: 7

    (e) `str.charAt(6)`
    return type: char, value: 'e'

    (f) `str.substring(1, 6)`
    return type: String, value: "ow ar"

11. (3 points) What is wrong with this piece of code? (Circle the error and tell me why it's wrong)

```
int count = 0;
    while (count < 100)
        if (count % 5 == 0)
            System.out.println(count);
            count++;
```

Answer: The right code is:

```
int count = 0;
while (count < 100) {
    if (count % 5 == 0)
        System.out.println(count);
    count++;
}
```

The code in the question will fall into an infinite loop.

12. (3 points) What is wrong with this piece of code? (Circle the error and tell me why it's wrong)

```
do {
    System.out.print("The programming language, "Java", ");
} while (false);
System.out.println("is named after the Java coffee");
```

Answer: The right code is:

```
do {
    System.out.print("The programming language, \"Java\", ");
} while (false);
System.out.println("is named after the Java coffee");
```

We must use "\" to print quotation marks.

13. (4 points) What is wrong with this piece of code? (Circle **two errors** and tell me why they're wrong)

```
public int absoluteValue(int num) {
    if (num < 0);
        return -num;
    else if (num > 0)
        return num;
}
```

Answer: The right code is:

```
public int absoluteValue(int num) {
    if (num < 0)
        return -num;
    else
        return num;
}
```

(1) There shouldn't be a semicolon after the "`if`" line;

(2) The case that `num` is equal to 0 is not considered.

Part 2. (20 points) Choose any TWO questions from question 14-16. Pay close attention to the point values of these questions. You may attempt 16 or 20 points depending on the questions you choose. You will receive partial credit for correct written reasoning procedures.

14. (8 points) Write the output of the method `doSomething()`.

```
public void swap(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
}

public void doSomething() {
    int a = 2, b = 3;
    a = b;
    b = a;
    System.out.println(a + "," + b);
    int c = 2, d = 3;
    int temp = c;
    c = d;
    d = temp;
    System.out.println(c + "," + d);
    int e = 2, f = 3;
    swap(f, e);
    System.out.println(e + "," + f);
}
```

Answer: The output is respectively:

```
3,3
3,2
2,3
```

15. (8 points) Write the output for the piece of code.

```
int a = 2, b = 2, c = 2;
for (int i = 0; i < 3; i++) {
    a += a;
    b *= b;
    c /= c;
}
System.out.println(a + ", " + b + "," + c + ".");
```

Answer: The output is:

```
16, 256, 1.
```

16. (12 points) Write the output of `test(1)`, `test(3)` and `test(5)`.

```
public void test(int k) {
    String t = "The quick brown fox jumps over the lazy dog";
    for (int i = 0; i < k; i++) {
        t = t.substring(t.indexOf(" ") + 1);
    }
    System.out.println(t.substring(0, t.indexOf(" ")));
}
```

Answer: The output is:

```
quick
fox
over
```

Part 3. (40 points) Choose any TWO questions from question 17-19. Question 17 and 18 have two possible versions. You can choose to implement the basic version or the advanced version. You will receive BOTH basic and extra credits if you implement the advanced version. You may attempt 35, 40, 45 or 50 points depending on the questions and versions you choose. Please indicate what version you are implementing. You will receive full credit for code that can be compiled and generate correct answers. You will lose points for either syntax or logical errors. You will receive partial credit for correct structures, statements or reasoning procedures.

17. (15 points for basic version) Write a method `divisors(int N)` that returns the **number of divisors** of a positive integer `N` that is passed in to the method. A divisor of an integer n, also called a factor of n, is an integer which divides n without leaving a remainder. For example, the number 100 has 9 divisors (1, 2, 4, 5, 10, 20, 25, 50, 100). Therefore `divisors(100)` should return 9. It is easy to test whether a number is a divisor of another number by the modulus operation. For example again, 5 is a divisor of 100 because 100 mod 5 is equal to 0; 6 is not a divisor of 100 because 100 mod 6 is equal to 4.

    (5 extra points for advanced version) The advanced version requires that the loop will execute for no more than `Math.sqrt(N)` times, where `Math.sqrt(N)` calculates the square root of `N` (with return type `double`). For example, your program should not loop for more than 10 times to calculate `divisors(100)`. (Hint: If 2 is a divisor of 100, then $100/2 = 50$ is also a divisor of 100.)

    Answer: The basic version is:

```
public int divisors(int N) {
    int count = 0;
    for (int i = 1; i <= N; i++) {
        if (N % i == 0)
            count++;
    }
    return count;
}
```

    The advanced version is:

```
public int divisors(int N) {
    for (int i = 1; i <= Math.sqrt(N); i++) {
        if (N % i == 0)
            count += 2;
        if (i == Math.sqrt(N))
            count--;
    }
    return count;
}
```

18. (20 points for basic version) Write a method `equalStrings(String a, String b)` that returns **whether String a and String b are exactly the same**. In other words, you are implementing `.equals()` method. For example, `equalStrings("abcd", "abcd")` should return `true`, and `equalStrings("abcd", "abce")` should return `false`. You are required to use only `.length()` and `.charAt()` methods in your method. You are not allowed to use `.equals()`, `.indexOf()`, `.lastIndexOf()` or `.compareTo()`. (Hint: You must compare the characters in the strings one by one.)

    (5 extra points for advanced version) The advanced version requires that the method will return `true` if the two strings only differ from **the spaces in the end**. For example, `equalStrings("abcd", "abcd ")` should return `true`. However, `equalStrings("  abcd", "abcd  ")` will still return `false`. You are not allowed to use `.trim()`.

    Answer: The basic version is:

```
public boolean equalStrings(String a, String b) {
    boolean result = true;
    if (a.length() != b.length()) {
        result = false;
    } else {
        for (int i = 0; i < a.length(); i++) {
            if (a.charAt(i) != b.charAt(i))
                result = false;
        }
    }
    return result;
}
```

(**Question 18 cont'd:**)

The advanced version is:

```
public static boolean equalTrimStrings(String a, String b) {
    boolean result = true;
    String LStr, SStr;
    if (a.length() >= b.length()) {
        LStr = a;
        SStr = b;
    } else {
        LStr = b;
        SStr = a;
    }
    for (int i = 0; i < LStr.length() - SStr.length(); i++) {
        if (LStr.charAt(i) != SStr.charAt(i))
            result = false;
    }
    if (LStr.length() != SStr.length()) {
        for (int i = SStr.length(); i < LStr.length(); i++) {
            if (LStr.charAt(i) != ' ')
                result = false;
        }
    }
    return result;
}
```

19. (25 points) Write a method `pi()` that returns the value of π, the ratio of a circle's circumference to its diameter. The value of π can be calculated by this infinite series:

$$\frac{\pi}{4} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} - \frac{1}{15} \cdots$$

Your method should use this infinite series to calculate the value of π. Because there are infinite numbers of terms in the series, you are required to compute **the sum of the first** 10000 **terms** in this series.

Answer: One possible version:

```
public double pi() {
    double qPi = 0;
    for (int i = 1; i <= 10000; i++) {
        if (i % 2 != 0) {
            qPi += 1 / (double) (2 * i - 1);
        } else {
            qPi -= 1 / (double) (2 * i - 1);
        }
    }
    return qPi * 4;
}
```

Another possible version:

```
public double pi() {
    double pi = 0;
    double term = 4;
    for (int i = 1; i <= 10000; i++) {
        pi += term;
        term *= -(double) (2 * i - 1) / (2 * i + 1);
    }
    return pi;
}
```

(**Question 19 cont'd:**)

The third possible version:

```
public double pi() {
    double qPi = 0;
    double divisor = 1;
    boolean odd = true;
    while (divisor <= 20001) {
        if (odd) {
            qPi += (1 / divisor);
            odd = false;
        } else {
            qPi -= (1 / divisor);
            odd = true;
        }
        divisor += 2;
    }
    return qPi * 4;
}
```

The fourth possible version:

```
public double pi() {
    double qPi = 0;
    double divisor = 1;
    for (int i = 0; i < 10000; i++) {
        qPi += (1 / divisor);
        qPi -= (1 / (divisor + 2));
        divisor += 4;
    }
    return qPi * 4;
}
```

This is the last page of the solution.