CS 1401, Exam #1, TR version

$\frac{80}{80} = \left(\frac{10}{10}\right)$

Date: Tuesday, September 23, 2014
Name (please type legibly, ideally in block letters): [redacted]

1. On September 23, 1846, astronomers discovered a new planet of Neptune. This was one of the first triumphs of computation: by using Newton's equations, scientists predicted where the new planet could be, and observations confirmed that there is indeed a planet at this location. The corresponding computations actively used logarithms tables.

- Explain who invented logarithms and how exactly logarithms help with computing; *hint:* they make multiplication faster by reducing it to addition.
- Describe one more event from history of computing.

10/10

- John Napier developed logarithms, which sped up computation because the properties of logarithms allow for multiplication to be described as addition: $\log(ab) = \log a + \log b$. Computers take advantage of this.

- George Boole developed what is now known as Boolean Algebra which is key in designing and understanding the operations of modern circuits.

2. For each of the following sequences of symbols, describe which can be valid Java identifiers and which cannot be; if you believe they cannot be, briefly explain why (e.g., "is a reserved word" or "does not start with a letter"):

- logarithm – Valid
- public – Not Valid: this is a reserved word
- 1846 – Not Valid: starts with a number
- 23September – Not valid: starts with a number.
- planet-Neptune Not Valid: Contains a dash

10/10

3. The following formula enables us to compute the gravitational force F caused by a planet of mass M on a planet of mass m at a distance r: $F = GMm/r^2$. Assuming that G, M, m, and r are already placed in the corresponding variables of type double, write a Java code statement for assigning the corresponding value to the variable F of type double. Explain, step-by-step, which arithmetic operations will be performed first, which next, etc., and trace the computations on the toy example when G = 2.0, M = 3.0, m = 1.0, and r = 3.0. Explain what happens if you simply write GMm in your Java code.

Java statement:

```
double F = G*M*m/(r*r);
```

Explanation step-by-step:

1. Because r*r is in parenthesis Java will evaluate it first.
2. After this there is only multiplication and division so it will go from left to right multiplying then dividing. So now it will multiply G*M
3. M will now be multiplied by the result of step 2
4. The result of step 3 will be divided by the result of step 1.

Example

→ G*M*m/(3*3)

→ 2*3*m/9

→ 6*1/9

→ 6/9

= 3

If you simply wrote GMm in the above Java statement, Java would search for a variable named GMm, but would find none thus giving an error.

4-5. Nowadays, we can simply use computers to perform multiplication. Write the main method which asks the user for his/her name, asks for the numbers that he/she needs to multiply, and then computes and prints the result. For example, if Bill Gates wants to multiply 3.5 by 2.0, your program should print the following message:

```
From: Computer
To: Bill Gates

Here is the result of your multiplication:
3.5 X 2.0 = 7.0.
```

Reminder: to read from the keyboard, you can define the reader as follows:

```
Scanner reader = new Scanner(System.in);
```

the header of the *main* method is:

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

```
public static void main (String [] args)
    {
      Scanner input = new Scanner (System.in);
      System.out.println ("Hello! Please enter your name.");
      String name = input.nextLine ();
      System.out.println ("Please enter the first number you want...
                                                    ... to multiply);
      double number1 = input.nextDouble ();
      System.out.println ("Enter the second.");
      double number2 = input.nextDouble ();
      double product = number1 * number2;
      System.out.println("From: Computer");
      System.out.println("To: " + name);
      System.out.println();
     System.out.println("Here is the result of your multiplication:");
     System.out.println(number1 + " x " + number2 + " = " + product
}
```

10/10

6. Suppose that in your computations, lengths are originally given in meters, and you need to change the units to centimeters. Since 1 m = 100 cm, we need multiply the length in inches by 100. Suppose that the length is stored in the integer variable *length*. Which of the two lines of code leads to a correct increase:

- length = length * 100;
- length = length * 100.0;

line 1 - Since length is an integer variable it can't store double values which will be the result of line 2.

If originally, before each of these two lines, we had length of 2 m, explain what will happen after each of these lines is implemented by Java. What is a clearer way (different from those above) to multiply the variable *length* by 100?

Clearer Way:

```
length *= 100;
```

- length = length * 100;

```
length = 2 * 100;
length = 200;
```

- length = length * 100.0;

```
length = 2 * 100.0;
length = 200.0;
length = 200;
```

At this point Java would give an error or cast the double to an int

7. Neptune is one of the largest planets in the Solar system, it is actually third largest by mass. Write a piece of code that decides which of the three given planets is the largest by mass. The names of three planets are stored in the variables *pla1*, *pla2*, and *pla3*, and the masses of these planets are stored in the variables *mass1*, *mass2*, and *mass3*. Use if-then statements to write down a piece of Java code that prints the name of the largest of the three planets.

Comment: There is no need to read anything, assume that all six variables have already been assigned values.

assuming they don't have the same masses:

```
if ( mass1 > mass2 && mass1 > mass3) {
    System.out.println(pla1);
}
elseif ( mass2 > mass1 && mass2 > mass3) {
    System.out.println(pla2);
}
else { System.out.println(pla3); }
```

what if equal?

8. A celestial body is called a planet if it is large enough and close enough to the corresponding star. Write down a Java statement that uses the known truth values *largeEnough* and *closeEnough* to assign, to a boolean variable *is_aPlanet*, true or false depending on whether it is a planet or not. Draw the truth tables for "and", "or", and "not". Use these truth tables to find the truth value for Pluto, which is close enough, but not large enough.

Java statement:

```
boolean is_aPlanet = largeEnough && closeEnough;
```