Name: _____

Student ID: _____

# CS133 Lab 1 – Number representation

## Objectives

- Understand how numbers, especially floating point numbers, are represented and processed in the processor.
- Manipulate floating point numbers at the bit level.
- Identify the differences between single-precision and double-precision floating point numbers.
- Observe the loss of precision and errors in floating point numbers.

## Prerequisites

- Java development environment and Java IDE on your machine.
- Review the IEEE standard floating-point representations

## Further Readings

- Steve Hollasch, IEEE Standard 754 Floating Point Numbers, Dec-2-2015, Access online http://steve.hollasch.net/cgindex/coding/ieeefloat.html

## Instructions

1. Create a Java project with a single main class.
2. Try the following two lines and write down the output below.
    a. `System.out.println(Float.floatToIntBits(123.5f));`
       Output:
    b. `System.out.println(Float.intBitsToFloat(0x42F70000));`
       Output:
3. Convert the following decimal real number to a binary real number:
   Decimal: 125.75
   Binary:
4. Review the IEEE single-precision floating point standard. What is the number of bits reserved for the sign, exponent, and fraction?
   Sign:
   Exponent:
   Fraction:
5. To represent this number in the IEEE standard single-precision format, we need to normalize that number first.

Normalized representation:
Mantissa:
Exponent:

6. Put the above number in the standard format by adjusting the mantissa and exponent:
   Sign bit:
   Adjusted exponent:
   Adjusted mantissa (fraction):
7. Provide two methods to verify your answer. (Hint 1: use the two methods in Step 2 above.) (Hint 2: You can use the system calculator in your OS.)

8. Repeat the above steps for the decimal number -0.2 (-ve number). What do you notice?

9. According to your understanding of the IEEE floating-point representation, how do we obtain the smallest +ve value using the method `Float.intBitsToFloat`? Is this a normalized or non-normalized number? Explain your answer.

10. What is the largest +ve value that can be represented in a single-precision floating point number? Please explain how to obtain this value using the method `Float.intBitsToFloat`.

11. In the following code snippet, what is the smallest value for the variable y that will cause the condition to evaluate to true? Please explain how the answer is obtained using the floating-point standard representation.

```
float x = 1.0f;
float y =  …;
if ((x+y) > x)
      System.out.println("not-equal "+(x+y));
```

12. Run the following code snippet and report the output.

```
float z = 1.0f / 0.0f;
System.out.println(z);
```

Output:

13. What is the bit representation of the value of z in the code snippet above?

14. Repeat parts 12&13 for the following expressions.

| Expression | Output | Bit representation |
|---|---|---|
| w = 0.0f / 0.0f | | |
| z - z | | |
| z * z | | |
| z / z | | |
| z * 0.0f | | |
| z * w | | |
| 0.3f – 0.3f | | |
| 0.3f – 0.2f – 0.1f | | |

15. Consider the following code snippet. Is there an assignment to x1 and x2 that causes the program to print “Case 4?” If yes, provide this assignment. If no, explain why not.

```
float x1 = …, x2 = …;
if (x1 < x2)
  System.out.println("Case 1");
```

```
else if (x1 > x2)
  System.out.println("Case 2");
else if (x1 == x2)
  System.out.println("Case 3");
else
  System.out.println("Case 4");
```

16. Will your answer in part 15 change if the type of x1 and x2 was `int`? Why or why not?