IN2009 Tutorial 1

This tutorial assumes that you have already carried out the Getting Started steps from Moodle, and that you attended Lecture 1 (or watched the recording). You will also need to refer to the Simple Stack Machine Specification in Moodle.

Remember:

- assemble: **`java Assemble <assembly-file-name>`**
- execute: **`java Exec a.out`**
- both: **`java Run <assembly-file-name>`**

`Run` (or `Exec`) with the `–debug` option allows you to run the SSM one step at a time. **Note**: make sure that you pass the `–debug` option to the SSM, *not* to the JVM:

☑ **`java Run -debug myprog.ssma`**

☒ `java -debug Run myprog.ssma`

1. Write an SSM assembly program which adds two integers together, and then prints the result using system-call number 3.
2. Write an SSM assembly program which calculates and prints the value of the expression:

   (3 * 7) + 9
3. Write an SSM assembly program which calculates and prints the value of the expression:

   3 * (7 + 9)
4. Write an SSM assembly program which behaves in the same way as this Java code:

```
 int x = 6;
int y = 13;
if (x < y) {
    System.out.println(y);
} else {
    System.out.println(x);
}
```

   You will need to use at least one kind of jump instruction. Note that in an SSM assembly program, statically allocated variables are declared at the end, in the .data section. For example, this program prints the value of variable z:

```
    // first push the memory address z
    push z
    // then load the value of z from memory
    load // now 94 is on top of the stack
    push 3
    sysc
    halt

.data

z:  94
```

5. Write an SSM assembly program which uses system-call number 1 to print the letter X.
6. Some SSM programs may run forever without ever halting. To terminate such a program you can press `Ctrl-C` at the command-line (hold down the Control-key and press C at the same time). Write an SSM assembly program which prints the letter X repeatedly, and never halts.
7. Write an SSM assembly program which prints X just 500 times, and then halts.
8. Write an SSM assembly program which prints your name. You could do this letter-by-letter, using character codes and system-call number 1, but a better way is to use a labelled string data definition in the .data section and system-call number 4. Something like this:

```
      // use system-call number 4 to print the string that
      // is stored at memory address myName

.data

myName: "Gary Cooper"
```

9. ***Challenge Exercise***. Write an SSM assembly program which prints your name using a string definition (as in exercise 8) and then prints it in reverse. Don't just define your name backwards as a second string in the .data section – that would be too easy! (Read the remarks about string representation towards the bottom of the SSM Specification document.) You might find it helpful to sketch a program in Java first, then translate to SSM assembly.