## COMP 141: Basic Semantics — Part 2

***Instructions:*** In this exercise, we are going to study environments, and assignment semantics.

(1) Consider the following program in C.

```
(1)     #include <stdio.h>

(2)     int a = 3;
(3)     int b = 5;


(4)     int main () {
(5)             printf("1. b=%d\n", b);
(6)             int b = a;
(7)             int a = 2 * b;
(8)             {
(9)                     printf("2. b=%d\n", b);
(10)                    int b = 1;
(11)                    {
(12)                            int a = b;
(13)                            int b = a;
(14)                            printf("3. a=%d\n", a);
(15)                            printf("4. b=%d\n", b);
(16)                    }
(17)                    printf("5. b=%d\n", b);
(18)            }
(19)            printf("6. a=%d\n", a);
(20)            printf("7. b=%d\n", b);
(21)            return 0;
(22)    }
```

(a) Define the environment at the end of each of the following lines: 7, 10, 13, 17, 18, and 22. For example, at the end of lines 3 and 7 the environment is as follows[1].

```
7:
b | a
```

(b) What would be the output of the program (i.e., the prints in standard output)?

[1] You can use your way of specifying stacks. No need to follow the syntax used here. Note that you should clearly specify the boundary between the items in each block. I suggest using `||`.

(2) In this exercise, we are studying value semantics and reference semantics in Java assignments. In Java,

- if the variables have primitive types (e.g., int, char, boolean, etc.) the assignment has value semantics.
- if the variables have object types (aka reference types), then the assignment has reference semantics.

However, we can also *imitate* value semantics for assignments in case the variables have object types. To this end, you can use `clone()` method, which creates a new cloned object that could be assigned to a variable [2].

Consider the following Java program.

```
(1)   public class C implements Cloneable {

(2)       int x;

(3)       C(int init) {this.x = init;}

(4)       void setX(int value) {this.x = value;}

(5)       int getX() {return this.x;}


(6)       public Object clone() throws
(7)                       CloneNotSupportedException {
(8)           return super.clone();
(9)       }

(10)       public static void main(String[] args)  throws
(11)                          CloneNotSupportedException {
(12)          C obj1 = new C(5);
(13)          int y = obj1.getX();
(14)          C obj2 = obj1;
(15)          C obj3 = (C) obj1.clone();
(16)          obj1.setX(3);
(17)          System.out.println(y);
(18)          System.out.println(obj2.getX());
(19)          System.out.println(obj3.getX());
(20)          System.exit(0);
(21)      }
(22)  }
```

(a) What is the semantics of assignment in each of the lines 3, 4, 12, 13, 14, and 15.

For example, in line 3, the assignment has value semantics, since variables have a primitive type (int).

(b) What would be the output of the program (i.e., the prints in standard output invoked in lines 17, 18, and 19)?

(3) Consider the block structure.

```
(1)   {
(2)           int a;
(3)           int b;
```

[2] Inherently, the assignment still has reference semantics, but cloning makes it act as if objects are copied.

```
(4)             ...
(5)             {
(6)                     char b;
(7)                     int c;
(8)                     long d;
(9)                     {
(10)                            bool a;
(11)                            int b;
(12)                            ...
(13)                    }
(14)                    ...
(15)                    {
(16)                            double c;
(17)                            short a;
(18)                            {
(19)                                    double b;
(20)                                    long c;
(21)                                    ...
(22)                            }
(23)                            ...
(24)                    }
(25)            }
(26)            ...
(27)    }
```

Define the environment in the lines marked with "···", i.e., the lines: 4, 12, 14, 21, 23, and 26.

(4) Consider the following code piece in a given PL.

```
x := 5;
y := x;
x := 6;
print(y);
```

What is the value bound to name `y`, after running this code piece, if

(a) assignment has a value semantics.

(b) assignment has a reference semantics.

(5) Consider the following program in C.

```
#include <stdlib.h>

int x = 1;

int main() {
        int y = 2;
        int *z;
        z = (int *) malloc (sizeof(int));
        *z = 3;
        z = 0;
        return 0;
}
```

(a) This program stores integer values `1`, `2`, and `3` in the memory before returning from `main` function. Specify the region of the memory each of these values are stored. The potential regions are: static area, stack, and the heap.

(b) Does this program generate garbage? (Yes/No) Explain the reason.