1. (4 points) Consider the following Java class.

```java
class Beta {
    public Beta b;
    public void test(int x) {
    }
}
```

For each of the declarations below individually, show how to write it, assuming it is the only statement placed in the box shown above, or explain why the declaration cannot be made.

(a) Declare a local variable b of type int with initial value 1

```java
int b = 1;     // ok. Local variable “b” will hide field “b” in class “Beta”
```

(b) Declare a local variable x of type Beta with initial value null

```
not possible – cannot hide the parameter “x” of method “test”
```

(c) Declare a local variable Beta of type int with initial value 1

```java
int Beta = 1; // ok. Class type “Beta” and variable “Beta” can coexist and are distinguished by the context of their use. But this is really poor practice!
```

(d) Suppose we place the statement `Beta b = b;` in the box. What goes wrong and why?

```java
Beta b = b;     // fails!
```

The variable declaration “Beta b” creates a local scope for the variable “b”. In this scope “b” is declared but not initialized. Thus Java warns that local variable “b” in the initializing expression may not have been initialized (in this case it definitely has not been initialized).
2. **(4 points)** In the space provided on the flip side of this page, write a machine code program (e.g. MIPS or ARM assembly or similar instruction set of your choice) that implements the following program fragment to compute the greatest common divisor $gcd(x,y)$ for positive integers $x$ and $y$.

```plaintext
while (x != y) {
    if (x > y)
        x = x - y;
    else
        y = y - x;
}
```

Assume initially the values of $x$ and $y$ are held in two general-purpose registers of your choice. On termination both these registers will hold $gcd(x,y)$. Strict adherence to assembly syntax is not required!

```plaintext
# input: registers $a0$ and $a1$ hold positive integers x, y
# output: registers $a0$ and $a1$ hold $gcd(x,y)$
loop: beq $a0$, $a1$, end
    blt $a0$, $a1$, else
if: sub $a0$, $a0$, $a1$
    b loop
else: sub $a1$, $a1$, $a0$
    b loop
end:
```
3. (4 points) Consider the following Java code.

```java
interface I1 { int x = 0; }

class T1 implements I1 { int x = 1; }

class T2 extends T1 { private int x = 2; }

class T3 extends T1 {
    int x = 3;
    void show() {
        // ...
    }
}

class Test {
    public static void main(String[] args) {
        new T3().show();
    }
}
```

For each of the four below, give an expression to be placed in the box in method `show()` that prints the value of the specified instance of `x`, or argue it cannot be accessed.

(a) `x` in `I1`.
   The declaration of `x` in interface `I1` is interpreted as static, hence `x` can be accessed as a class member using
   
   ```java
   I1.x // x in interface I1
   ```

(b) `x` in `T1`.
   `T3` inherits `T1`, hence `x` in `T1` can be accessed as `super.x` from `T3`, or by (down)casting the `T3` instance to a `T1` instance.

   ```java
   super.x or ((T1) this).x // x in superclass T1
   ```

(c) `x` in `T2`.
   `T3` does not inherit `T2`, so there is no instance of `x` in `T2` within `T3` to be accessed. The class `T2` is declared in the same (anonymous) package, so it is possible to create an instance of `T2` but we cannot access `x` in that instance since it is private.

(d) `x` in `T3`.
   It can be accessed simply as `x` or `this.x`

   ```java
   x or this.x // x in T3
   ```