Part D. Name three benefits of including functional style programming in Java

- Programs are more compact: easier to write, and easier to read/understand
- Programs are thread-safe
- Easier to demonstrate correctness of functional programs
- Easier to test; less likely that a test of a subroutine will fail tomorrow if it passed today since there are no side effects

Part E. Express the functions defined below using Church's lambda notation:

i.	f(x) = x + 2x2	=>	λx.x + 2x2
ii.	g(x,y) = y – x + xy	=>	λxy.y – x + xy
iii.	h(x,y,z) = z – (x + y)	=>	λxyz. z – (x + y)

Part F. For each lambda expression below, name the parameters and the free variables.

i.

```
//parameters are marked in blue; free variables are marked in red
Runnable r = () ->
 {
     int[][] products = new int[s][t];
     for (int i = 0; i < s; i++) {
           for(int j = i + 1; j < t; j++) {
                 products[i][j] = i * j;
           }
     }
 }
```

ii.

```
//parameters are marked in blue; free variables are marked in red
BiFunction<Double, Double, Double> f = (double u, double v) → ∫ₐᵇ cosux sinvx dx
```

(Note: the right hand side of the "→" is mathematical notation, not Java, but it can be converted to a large block of Java code having the same free variables. See lecture code to review the BiFunction functional interface.)

iii.

```
//parameters are marked in blue; free variables are marked in red
Comparator<String> comp = (s, t) ->
 {
     if(ignoreCase == true) {
           return s.compareToIgnoreCase(t);
     } else {
           return s.compareTo(t);
     }
 }
```