CS2030 Programming Methodology
Semester 2 2018/2019

27 March – 29 March 2019
Tutorial 7
Java Streams and Functional Interfaces

1. Given the following class `A`.

```
class A {
    int field;
    void method() {
        Function<Integer, Integer> func = x -> field + x;
    }
}
```

   Model the execution of the program fragment:

```
A a = new A();
a.method();
```

   In particular, focus on the use of the *stack* and *heap* memory.

2. Suppose we have the following lambda expression of type `Function<String, Integer>`:

```
str -> str.indexOf(' ')
```

   (a) Write a `main` method to test the usage of the lambda expression above.

   (b) Java implements lambda expressions as anonymous classes. Write the equivalent anonymous class for the lambda expression above.

3. Complete the method `and` that takes in two `Predicate` objects `p1` and `p2` and returns a new `Predicate` object that evaluates to `true` if and only if both `p1` and `p2` evaluate to `true`.

```
Predicate<T> and(Predicate<T> p1, Predicate<T> p2) {
```

4. Write a method product that takes in two `List` objects `list1` and `list2`, and produce a `Stream` containing elements combining each element from `list1` with every element from `list2` using a `BiFunction`. This operation is similar to a Cartesian product.

```
public static <T,U,R> Stream<R> product(List<? extends T> list1,
        List<? extends U> list2,
        BiFunction<? super T, ? super U, R> func)
```

For example, the following program fragment

```
List<Integer> list1 = new ArrayList<>();
List<Integer> list2 = new ArrayList<>();

Collections.addAll(list1, 1, 2, 3, 4);
Collections.addAll(list2, 10, 20);

product(list1, list2, (str1, str2) -> str1 + str2)
    .forEach(System.out::println);
```

gives the output

```
11
21
12
22
13
23
14
24
```

5. Write a method that returns the first n Fibonacci numbers as a `Stream<BigInteger>`. The `BigInteger` class is used to avoid overflow.

   For instance, the first 10 Fibonacci numbers are $1, 1, 2, 3, 5, 8, 13, 21, 34, 55$.

   Hint: It would be useful to write a new `Pair` class that keeps two items around in the stream.