1. An advantage of pass-by-name (Jensen’s Device)In the lecture, we argued and demonstrated that pass-by-name can lead to some rather unexpected behaviour, and that most contemporary languages don’t use it. Instead, passing references by value (as done in Java) is preferred.

   However, there are some elegant things we can do if we use the pass-by-name paradigm. For example, we can pass expressions into a procedure so that the expression can be repeatedly evaluated. Consider a calculation of the type “given an integer series a_k, calculate its sum $\sum_{k=l}^{u} a_k$

   Q 1.1: Write a Java method to accomplish this (hint, try to use the stream knowledge you’ve learned with HW 2 !!).

   Now, see the following code, which would have worked correctly with pass by name:

```
public int sum(int k, int l, int u, int ak) {
      int s = 0;
      for (k = l; k <= u; k++)
            s = s + ak;
      return s;
}
```

   With pass-by-name, to obtain the sum of the first 100 terms of an integer array arr would be called as follows:

```
sum(i, 1, 100, arr[i]);
```

   The terms `i` and `arr[i]` are being passed by name. As a result, the variable `ak` in the above method gets automatically re-evaluated based on the side-effect on the variable k in each iteration of the for loop. This is a simple example, but using the same technique, we could sum up much more complicated expressions, e.g., $\sum_{k=l}^{u} a_k(k + a_k)$

   The expression as a whole could be passed around because it will be automatically re-evaluated in every iteration.

   Q 1.2: This behavior can be mimicked in Java by using streams and lambda expressions for lazy evaluation. Write a Java method to do this.

2. A disadvantage of Pass-by-Name

   Consider the following standard way of writing a method to swap two bindings:
   The expression as a whole could be passed around because it will be automatically re-evaluated in every iteration.

```
// assume the generic parameter is defined for the class
public void swap(T x, T y) {
      T temp = x;
      x = y;
      y = temp;
}
```

   Q 2.1: What will happen in the following code with (a) pass-by-reference, and (b) pass-by-name?

```
      String s1 = “the first string”;
      String s2 = “the second string”;
      swap(s1, s2);
```

   Q 2.2: What will happen in the following code with pass-by-name?

```
      public static void main(String[] args) {
            int[] numbers = {1,2,3,4,5};
            int i = 2;
            swap(i, numbers[i]);
      }
```