CMPT 470/816 **HOME EXERCISE 7**

STREAMS

Within this problem, you will explore non-strict computation by creating streams in Scala.

In pursuing this home exercise, as in the last one, you may wish to recall that

- given an existing Stream[T] stream and a value v of type T, one can append an element on to that stream with the following syntax: *v #:: stream*
- Assuming that stream contains at least one element
  - Ahe first element of that stream s can be accessed by the following expression: *s.head*
  - The tail of that stream (the stream consisting of all elements but the first) can be accessed by the following expression: *s.head*

Create a function (parameterized by type T) which, given two infinitely long (unbounded) streams A and B of type T, returns a stream whose successive elements alternate are drawn from successive elements of A and B (in strict alternating succession between these two streams). That is, all elements of the returned stream with even numbered indices (treating the first index as 0) should represent the successive elements of A, and all elements of the returned stream with odd indices should be drawn from B. In other words, the first element of the returned stream should be the first element of A, second element of the returned stream should be the first element of B, the third element of the returned stream should be the second element of A, the fourth element of the returned stream should be the fourth element of B, and so on.

Due in class February 7, 2017