Review team A

The basic functionality of converting HM to CR is there. Converting to ANF is done by introducing lets at a low level with names depending on the terms and transforming the applications. The ANF HM datatype however is not really necessary as it has the same structure as the original HM datatype. The attribute grammar was not used to full extent in the conversion, because you could have kept track of the let bindings in an attribute instead of doing explicit recursion.

Primitive functions can also be handled by your solution and seem to work fine.

The Boolean and list types are built in to the compiler. The Boolean types work fine, but constructing lists doesn’t always work, because the head function applied on a take on a finite list returns a thunk instead of an actual value. Part of it seems to be there, but it still has some errors.

If expressions and recursion are executing as expected.

Laziness is not supported as explained in the README.

The overall impression of your code is that some code is hard to read, because it is hardcoded, like the references in the CR AG especially in BuildIn.ag where you define the isNil by referring to the other global bindings with plain numbers. Those numbers are repeated in the construction of the actual CR module, which could have been nicer. Also the removeDup in the ANF conversion is rather odd, why would you introduce duplicates in the first place? The rest of the implementation is nice and functional.

Grade: **8**