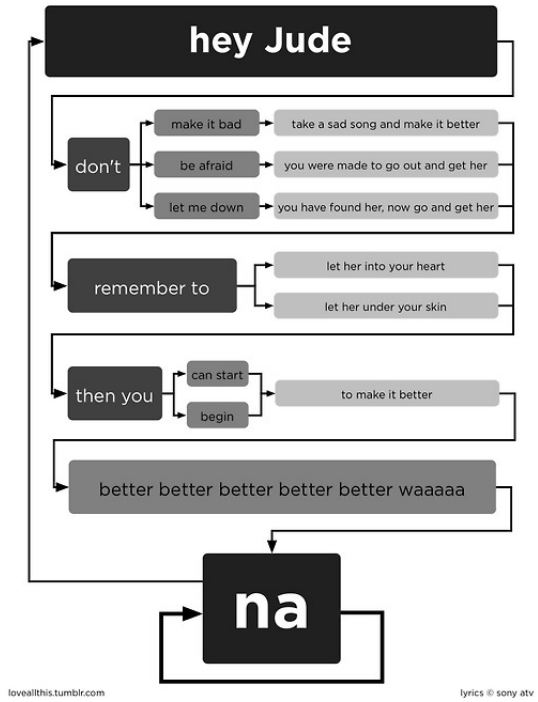
COMPSCI 3MI3 : Assignment 2

Fall 2021

Nicholas Moore

1. (4 points) **Question 1 : Hey Jude**
   Consider the following diagram:



   Write an EBNF grammar which completely describes the grammar of the song "Hey Jude" by the Beatles, as given in the diagram above.

2. (6 points) **Question 2 : Syntax Rapid Fire!**
   Write EBNF grammars for the following.

   - Identifiers in Python
   - Identifiers in Haskell
   - For loop in C. You don't have to write grammar for statements or expressions.

   Note: This question expects you to do a little research to find out what characters are valid in Python and Haskell identifiers.

3. (10 points) **Grammar in More Depth**
   Define the following list of terms in Python using the inference rule style we discussed in lecture. You do not need to define any terms other than the ones given.

   - for loop
   - while loop
   - Assignment statement
   - Slicing operation with all start and end point, as well as interval specified.

- function declaration

4. **Implementation of UAE in Haskell**

   (a) (3 points) **Defining Our Terms**
   Using Haskell's Type System, create an implementation of Untyped Arithmetic Expressions, as described in lecture. You should implement this language as a custom data type. Please have whatever types you define derive the Eq and Show typeclasses. *You don't have to implement any semantics or type checking.* These are topics we'll be covering later in the course, and you'll get plenty of practice in later assignments.

   Note: Transcribing the term names directly from the textbook isn't going to work, so you'll have to modify them slightly to avoid conflicts with some Haskell keywords like `True` and `False`. You are free to choose whatever modifications you like to the term names, just make sure your terms are still understandable as the originals.

   (b) (3 points) **Writing Untyped Arithmetic Expressions** Write Haskell functions which encode the following arithmetic expressions. These functions should constants of the term type.

$$\texttt{succ}\ (\texttt{succ}\ (\texttt{succ}\ 0)) \tag{1}$$

$$\texttt{succ}\ (\texttt{pred}\ (\texttt{iszero}(\texttt{succ}\ \texttt{true}))) \tag{2}$$

$$\texttt{if}\ (\texttt{iszero}\ (\texttt{succ}\ (\texttt{pred}\ 0)))\ \texttt{then}\ (\texttt{pred}\ \texttt{false})\ \texttt{else}\ (\texttt{succ}\ (\texttt{succ}\ (\texttt{succ}\ (\texttt{succ}\ 0)))) \tag{3}$$

   Once you have written these expressions, load your Haskell file into ghci (because who needs the hassle of adding a main function). If you've done everything correctly, this file should typecheck, and you should even be able to write arithmetic expressions in ghci using your brand-new term datatype.

   (c) (4 points) **Wait, Isn't that an Error?** While syntactically correct, the expressions in the previous question contained some semantic errors. Find all such errors, and explain why the error is an error.