# Design Document

Blake Le, Cameron Kocher, and Robert Yelas

1. **Introduction**

    - What is the name of your language?
    - What is the language's paradigm?
    - (Optional) Is there anything especially unique/interesting about your language that you want to highlight up front?

Our language is called **does this sound bad**, also known as **dtsb**. It uses an imperative paradigm, and is intended to be simple and easy to use, with uncomplicated syntax. We named our language after an email shared by our friend that we mercilessly memed into the ground:

Thank you for your email!

In terms of taking the midterm and final on the same day, I would actually prefer this if at all possible. Having an additional 2 days to study for a comprehensive test such as the final will likely not impact my scores in any significant way. I do completely understand that this may cause issues on your end, and will just take the final during finals week if this is not possible.

Best,

<name omitted>

does this sound bad

2. **Design**

    - What *features* does your language include? Be clear about how you satisfied the constraints of the feature menu.
    - What *level* does each feature fall under (core, sugar, or library), and how did you determine this?
    - What are the *safety properties* of your language? If you implemented a static type system, describe it here. Otherwise, describe what kinds of errors can occur in your language and how you handle them.

When completed, our language will include the basic building blocks of a programming language: standard data types, declarable and mutable variables, if-else conditionals, while and

for loops, and functions that are able to accept arguments. We'll also implement strings, tuples, and lists, as well as built-in methods that operate on these data types. Of the aforementioned planned features, we have already implemented the following: standard data types, if-else conditionals, tuples, basic list operations, strings, and string manipulation functions.

**Core Features:** data types, variables, if-else conditionals, while loops, functions
**Syntactic Sugar:** for loops, tuples
**Library:** strings, string manipulation functions

We have not yet implemented any safety properties and are relying solely on Haskell's error-handling for now. We currently have no plans to implement a static type system to allow for error checking prior to running.

3. **Implementation**

    - What *semantic domains* did you choose for your language? How did you decide on these?
    - Are there any unique/interesting aspects of your implementation you'd like to describe?

For our semantic domains, we chose integers and booleans for now because of their simplicity and their wide range of uses. For the final version, we plan to also implement characters and strings as well to cover most bases of coding.