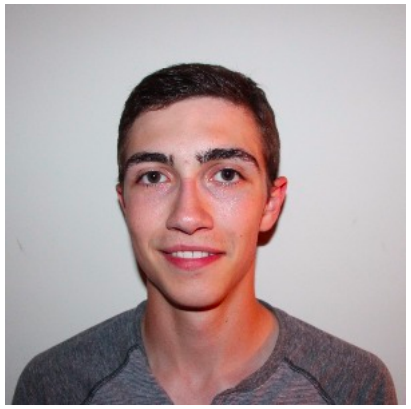
Design-ReSOARpe

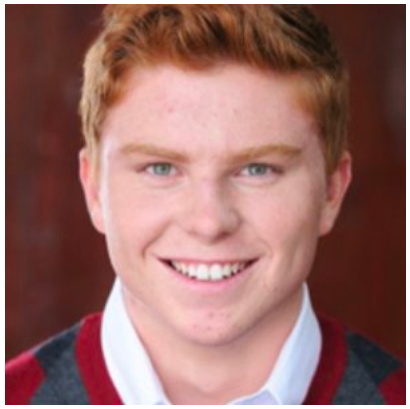
A Challenge Problem

Collaborators

Ramzi Talhouk



James Medlin



Goal: Soar Knows "Fundies"

CS 2500 | FUNDAMENTALS OF COMPUTER SCIENCE 1

The design recipe

DATA	FUNCTIONS
① **Data definition** *Give the data definition a name, and state the set of values that are part of it.*	① **Signature** *Give the name of the function, the argument types that it expects as input, and what type it returns.*
② **Interpretation** *State how values should be interpreted, covering each field/clause if there are multiple of them.*	② **Purpose statement** *Describe in one sentence what the function does. This should be roughly the length of a tweet.*
③ **Examples** *Provide a set of representative examples, building up complex examples iteratively.*	③ **Tests** *Provide a set of representative tests, covering different kinds of input and different behaviors.*
④ **Template** *Provide a template for functions that accept this data definition as input.*	④ **Code** *Starting with the template for the input data definition, write the code for the function.*

CARD 1

Example

- A bank account contains a sequence of transactions, each with a date, notes about the transaction, and amount deposited/withdrawn.

- Design a function that accepts a bank account and returns the current balance of the account.

Data Design

```
(define-struct transaction [date notes amount previous])

; A BankAccount is one of:
; - #false
; - (make-transaction String String Number BankAccount)
; Interpretation: Represents a bank account history
; - #false represents the beginning of the account
; - date is the date of the transaction in YYYY-MM-DD format
; - notes is notes about the transaction
; - amount is the amount deposited (if positive) or withdrawn (if negative)
; - previous is the BankAccount before this deposit

(define ACC-0 #false)
(define ACC-1 (make-transaction "2018-01-01" "Paycheck" 500.00 ACC-0))
(define ACC-2 (make-transaction "2018-01-07" "Dinner" -45.50 ACC-1))

#;
(define (bankaccount-temp acc)
  (cond
    [(boolean? acc) ...]
    [(transaction? acc) ... (transaction-date acc)...
                        (transaction-notes acc) ...
                        (transaction-amount acc) ...
                        (bankaccount-temp (transaction-previous acc))]))
```

Function Design

```
; current-balance : BankAccount -> Number
; Computes the current balance of the account

(check-expect (current-balance ACC-0) 0)
(check-expect (current-balance ACC-1) 500)
(check-expect (current-balance ACC-2) 454.50)

(define (current-balance acc)
  (cond
    [(boolean? acc) 0]
    [(transaction? acc) (+ (transaction-amount acc)

                           (current-balance (transaction-previous acc)))]))
```

Interesting Aspects

- Data design requires broad knowledge of the world
- Data/function design require complex, intermingled reasoning
  - Choice of data model influences processing and vice-versa
- Desirable to have a multi-step, interpretted process
  - Useful for modeling problem difficulty, understanding student conceptual deficiencies
- Potential for interactive question-answering in the case of problem ambiguity

Progress So Far

- Tried to simplify… unclear how to do so
  - Example: supplying input/output tests converts to function-induction problem, which is hard and undesirable
- Tried to come up with simplified input format (to avoid NLP)… hard cliff to trivialization
- Performed pair-obfuscation to tease out assumptions and processes
  - Have a small dataset

Evaluation

- Interesting challenge problem for Soar
  - Requires large amounts of world knowledge, complex reasoning, explanation
- Real applications inside/outside the classroom

- Tough (nugget of) coal to crack
  - Hard to find opportunities for simplification