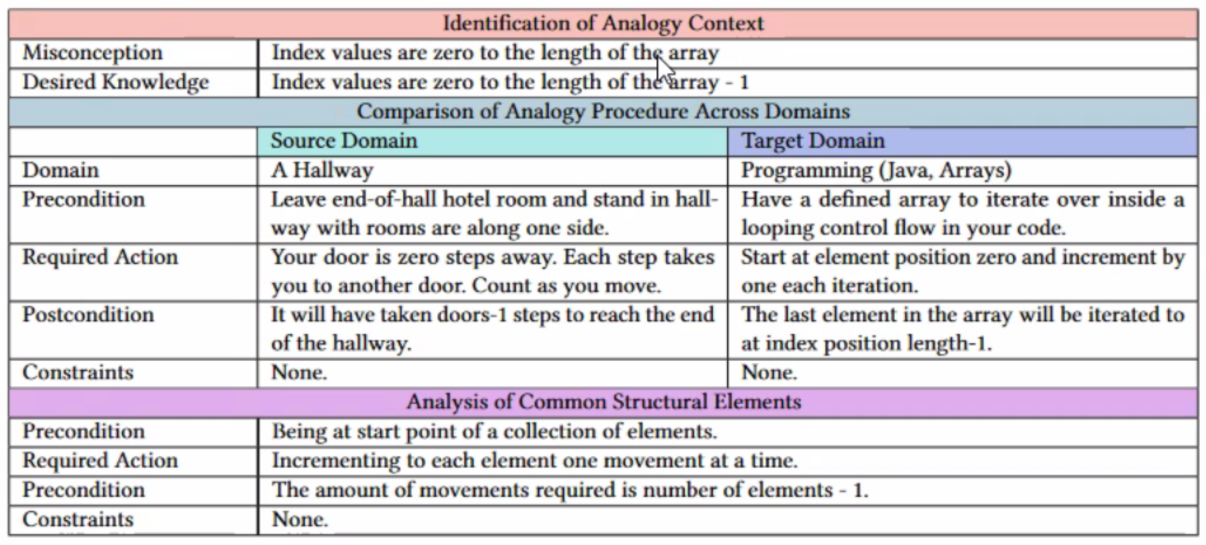
Team name: Programming Analogies
Date, time, and duration: January 24, 2023 4PM EST for 30 minutes
Location: Zoom meeting
Attendance

- Dr Bettin (Scientist)
- Emilie Rummer
- Ethan Jones
- Jack Grant
- Josh Staples
- Kevin Kulich

Discussions and Decisions: Open-panel discussion for further application requirements and design ideas. All questions were clarified, with corresponding notes below in the "General Notes" idea.
General Notes

- Biggest priority: displaying analogies, then searching through analogies, then creating analogies
- Make analogies based on potential problems/misconceptions, consider how you'd explain things to other people (more info in Dr Bettin's dissertation)
- Analogy comparison should be able to pin different parts of analogies and use collapsable columns
- Users should only be able to edit/delete their own analogies (with the exception of admins)
- If we add in favoriting analogies, users should be able to add notes/essentially fork other analogies
- Possibly have a public student view (search/browse only)
- Analogy creation should have different sections and mirror information that's already been filled in
- Analogy creation may use different pages but could also be a two column layout, it's up to us to design that
- The analogies shouldn't be exclusive to more beginner programming concepts, they should also include upper level concepts

Table 1: Design Worksheet Example: Final Array Position at Length-1—A Hallway



Identification of Analogy Context		
Misconception	Index values are zero to the length of the array	
Desired Knowledge	Index values are zero to the length of the array - 1	
Comparison of Analogy Procedure Across Domains		
	Source Domain	Target Domain
Domain	A Hallway	Programming (Java, Arrays)
Precondition	Leave end-of-hall hotel room and stand in hallway with rooms are along one side.	Have a defined array to iterate over inside a looping control flow in your code.
Required Action	Your door is zero steps away. Each step takes you to another door. Count as you move.	Start at element position zero and increment by one each iteration.
Postcondition	It will have taken doors-1 steps to reach the end of the hallway.	The last element in the array will be iterated to at index position length-1.
Constraints	None.	None.
Analysis of Common Structural Elements		
Precondition	Being at start point of a collection of elements.	
Required Action	Incrementing to each element one movement at a time.	
Precondition	The amount of movements required is number of elements - 1.	
Constraints	None.	