

# Solving Complex Problems

Malakai Spann  
April 2025

# Content

01.

Solution Overview

02.

This Problem

03.

"The Problem"

04.

Asking Questions

# Solution Overview

Language Chosen: Zig

Time to Complete: ~25 hours

Key Features and Concepts:

- Custom String Implementation
  - Memory Management
- Custom HTML Parser Implementation
  - Recursion
- Custom Docker Image
  - Open to Extension, Closed to Modification



Source: [Docker Media Page](#)



Source: [Zig Language Repository](#)

# What's the Problem

## Use the Given Language (& nothing but)

1. Only use the assigned language, its standard library, and built-in features

## Get Something from the Web

1. Scrape the content of a Wikipedia Page.
2. Extract the HTML from the scraped content.

## Do Something with It

1. Extract the table data out of the HTML.
2. Output the extracted table data in CSV format.

## Containerize It All

1. Create an image capable of building and running the software.
2. Configure a container using that image to automatically run the software

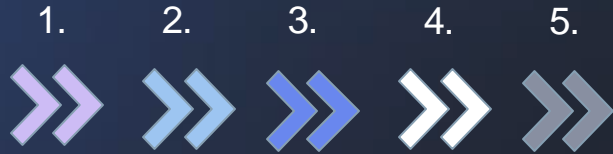
# The Problem with “Problems”



Problems are often presented in their largest, most intimidating form.

# Perception vs Reality

How Problems are Introduced



What Problems Actually Are



Okay, Okay. How?



# The Art of Asking Questions

## What Do You Know that You Know?

This list will often be the shortest.

These points can also give you an idea of what you don't know.

## What Do You Know that You Don't Know?

This list may seem to be the shortest, but, it will constantly grow.

## Have You Encountered Something Similar?

The things you already know could be helpful in understanding what you don't.

Think of these points as starters for new questions.

Then, very important, Google it!



# Example Flow

I Know...

"I'll need to access the information from the website in my program."



I Don't Know...

"If <language x> has a way to get information from the internet."



Crossover

"In <language y>, there's a standard module that has a function capable of making 'requests' to the."



"How <language y> makes requests to the internet."

Google: "<language x> internet requests"

# Example Flow (cont.)

I Know...

"I can make HTTP requests to get information from the internet."



I Don't Know...

"If <language x> has a standard HTTP 'library'."

"Where documentation for <language x's> is."



Crossover

"In <language y>, the standard library is documented on the language's web page".

Google: "<language x>  
documentation"

# Example Flow (cont.)

I Know...

"<language x> has a HTTP library with standard HTTP methods."



I Don't Know...

"How to use the HTTP methods of the library."


"How <language x> handles data types and data storage."



Crossover

"Other languages have functions that return data with certain types or made available in data structures".

Google: "<language x> data types" or "<language x> classes"

The background features a dark blue gradient. At the top center, there is a series of nested, downward-pointing triangles. In the bottom left corner, there are several parallel diagonal lines. In the bottom right corner, there are several concentric circles.

ALWAYS RELATE YOUR  
UNKNOWN TO SOMETHING  
YOU ALREADY UNDERSTAND.

# Final Thoughts



## Remain Curious!

Each answer builds the foundational knowledge to answer new questions.

## Approach

- Determine knowns and known unknowns
- Formulate questions
- Rinse and Repeat

### Key takeaway 1

Turn each complex "problem" into a small collection of simple problems.

### Key takeaway 2

The solution is never as important as the understanding gained from it.

### Key takeaway 3

Draw from your experience. Lots of problems are conceptually similar to one another.

# Stay In Touch



MalakaiSpann.com

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**

Please keep this slide for attribution