# GEL Specification Doc

## Step 1: Add Exam Generator Prompt to your AI agent

(Copy exact text below)

**Purpose:** Generating Excel 365 based exam questions

**Details:** Write an Exam problem based on the problem statement in `++X`. Generate a `LAMBDA` that solves the problem, but it is sprinkled with `++Y UNKNOWNS` that students need to solve. Same `UNKNOWN` should not be used multiple times. Once the `UNKNOWNS` are filled in, the requirements must be implemented correctly for any input. The possible correct answers for the `UNKNOWNS` must be small (¡10). Use PascalCase for all names throughout the `LAMBDA`. If a name ends in a number put an underscore in front of the number, e.g., `X_1` to avoid confusion with `X1` which exists in the grid.

In addition to the `LAMBDA` with gaps, provide a correct answer for all `UNKNOWNS` and explain how the correct `UNKNOWN` was obtained.

**Template Parameters:**

`++Y (Number of UNKNOWNS) = 10`

**Shape of UNKNOWNS:** Excel 365 formulas with nesting level at most 1, e.g., `COUNTIFS(W, P)` or `LAMBDA(acc, y, acc+y)` or 4 or `TotalShare`. About half of the `UNKNOWNS` should be level 0 (no nesting) and the other half level 1.

**Solution shape:** `LAMBDA` with `LET` and built-in debugging showing all intermediate results with `HSTACK` using the Helper Array Debugging approach (HAD) described below in `++HAD_Instrumentation`.

Provide a correct answer for all `UNKNOWNS` and explain how the correct `UNKNOWN` was obtained.

### ++HAD_Instrumentation

#### Boilerplate Code Generation for LAMBDAs with a top-level LET

This is a task for precise boilerplate code generation for Excel 365. The generated code helps debug `LAMBDAs` using the Helper Array Debugging (HAD) technique. We give a `LAMBDA` template and show the corresponding translation, which we call `HAD_instrumentation`. `HAD_instrumentation` follows the Principle of Least Knowledge: The main functionality in the `INPUT***` does not need to know how the debugging is achieved. The debugging formulas are cleanly separated from the main functionality shown in `OUTPUT***`. The output must follow the Principle of Least Knowledge.

`HAD instrumentation` does not allow the `echeck` function to be changed in any way. Use `VALUETOTEXT` as shown below!! Do not delete commas anywhere in the boilerplate code.

**INPUT*****

```
LAMBDA(a_1,a_2,a_3, ... LET(
  m_1,formula_1,
  m_2, formula_2,
  m_3, formula_3,
  ...
  Show)
)
```

**OUTPUT*****

```
LAMBDA(a_1,a_2,a_3, ... LET(
  m_1,formula_1,
  m_2, formula_2,
  m_3, formula_3,
  ...
  COMMENT_1, "Debugging Section",
  echeck, LAMBDA(value,IFERROR(value, "ERROR " & VALUETOTEXT(value,1) )),
  Show,IFERROR(HSTACK(
    "a_1",echeck(a_1),
    "a_2",echeck(a_2),
    "a_3",echeck(a_3),
    ...
    "m_1",echeck(m_1),
    "m_2",echeck(m_2),
    "m_3",echeck(m_3),
    ...
  ),""),
  Show)
)
```

End of Exam generator

Also note, we only want questions with iterative solutions, not recursive.

**Sample Problem Definitions**

**Sample 1: Billing Data Requirement**

Create a dynamic formula that calculates the water bill for a customer based on the rules below per 1,000 gallons.

- Up to and including 30,000 = $1.50 (i.e., 1000 gallons cost $1.5)
- Above 30,000 = $2.50 (i.e., 1000 gallons cost $2.5)

The lower price only applies if the total water usage is below the threshold. You cannot use conditional functions, such as IF or IFS, to solve the problem.

**Sample output lambda with unknowns:**

```
=LET(
  use, TAKE(CustT[Usage], 5),
  lookup_res, UNKNOWN_1(use, UNKNOWN_2, UNKNOWN_3, "error", UNKNOWN_4),
  water_bill, UNKNOWN_5 * lookup_res,
  debug, IFERROR(HSTACK("use", use, "lookup_res", lookup_res, "water_bill", water_bill), ""),
  debug)
```

**Sample 2: SwapFirstLast Requirement**

Write a function `SwapFirstLast(column)` in Excel 365, which switches the first element of the column with the last element. Use `LAMBDA(column, LET(...))`. The input column must contain at least one cell.

**Sample output lambda with unknowns:**

```
= LAMBDA(column,
  LET(
    n, ROWS(column),
    indices, UNKNOWN_1(n),
    swappedIndices, IF(UNKNOWN_2 = 1, n, IF(UNKNOWN_2 = n, 1, UNKNOWN_2)),
    result, UNKNOWN_3(UNKNOWN_4, swappedIndices),
    debug, HSTACK(
      "n", n,
      "indices", indices,
      "swappedIndices", swappedIndices,
      "result", result
    ),
    debug
  )
)(SEQUENCE(4))
```

**Inputs to Expect**

You can either give `++X` or `++NEWQUESTION`

**Output Details:** Based on the above instructions, you should return a `LAMBDA` with `UNKNOWNS` such as `UNKNOWN_1`, `UNKNOWN_2`, `UNKNOWN_3`, etc. After students fill in the unknowns, they can check their work against the provided solutions.

If given `++X`, your response should include:

- `Problem Definition`
- `LAMBDA problem with UNKNOWNS`
- `SOLUTION` with filled-in values
- `Table of UNKNOWNS`
- `SOLUTION without HAD`

If given `++NEWQUESTION`, you should generate a similar structure to `++X`, following all GEL instructions and samples.

Please wait for the command `++X` or `++NEWQUESTION` in the upcoming prompts!

## Step 2: Generate Exam Questions

Start the prompt using:

`Now comes the ++X: ++X = <PROBLEM NAME> Requirement`

`<Problem details>`

## Sample 1:

`++X = Billing Data Requirement`

Create a dynamic formula that calculates the water bill for a customer based on the rules provided earlier. You cannot use conditional functions such as IF or IFS to solve the problem.

## Sample 2:

When you do `++NEWQUESTION`, the AI agent will generate a new question for you following the GEL instructions.

**Note:** Use this to generate questions to practice for your exam. The AI agent may give incorrect solutions sometimes—don't worry! You have been well-prepared to debug with AI agents, and always remember to add debug when unsure!