1. DATA9: Binding Semantics, Parameter Passing, Pass By Need
    a. This would print “bar baz 1” since we are passing by value and the original a is never changed
    b. This would print “bar baz 4” since a’s memory address is being referred to and the memory value of a is being changed
    c. Would still print “bar baz 1” because in foo() a is being pointed to the object of 3, meaning that the original a still points to 1
    d. We would print “bar 1” because we never actually use baz and the modification on foo() and bar() are only done on local copies

2. FUNC1: Default Parameters
   You could look to use optional parameters and you can use the function itself to check if the optional parameters were actually passed in and should be processed. Additionally you could use positional parameters still, but use a placeholder such as Default or something if you want to change y and z but say you want to keep x the same as default.

3. FUNC2: Lambdas, Closures
   Closures in JS allow inner functions to "remember" and access the variables in the scope where they were created. In this case, the lambda function retains access to the counter in main, enabling it to increment the counter and produce the expected output. This behavior is fundamental to JavaScript’s functional programming capabilities, as it enables data sharing across different function scopes in a controlled way.

4. FUNC3: Error Handling, Optionals, Exceptions
   It makes more sense to use the optional because for the user, you either have to see if you returned a nullptr or you need a try and catch block which could get expensive and complicated and for a job as easy as searching an array you really do not need to go through all of that hassle. It’s a lot easier just to search if you got a nullptr from running this function.

5. FUNC4: Results, Optionals, Errors, Exceptions
    a. It is probably good to use assert() since we don’t actually need to know why the URL failed and all other methods give you details as to why the statement failed. Also it’s good to exit our program immediately after because if you can’t even get into the website there’s no need to run the rest of the program.
    b. Probably exceptions so we can catch it and process it before the error affects the rest of the rocket
    c. Probably optional, since we don’t need an exact error condition. We already know if the value is not between 1 - 1000 we get an error. Plus optional does have a

valid option so we can use that if we get a valid number to keep the program going

d. Likely a result object, since we get valid results most times and results can be valid. Otherwise it is likely a good idea to know where our API is failing and the result can return a specific error object so it's good to use result.

6. FUNC5: Exceptions

bar(0):
catch 2
I'm done!
that's what I say
Really done!

bar(1):
catch 1
hurray!
I'm done!
that's what I say
Really done!

bar(2):
catch 1
hurray!
I'm done!
that's what I say
Really done!

bar(3):
catch 3

bar(4):
hurray!
I'm done!
that's what I say
Really done!