1. What is the syntax for declaring a function in your language?
   1. The syntax for declaring a function in Fortran is as follows:
      1. Determine what data type you need for this function (REAL, INTEGER, etc)
      2. You may need to include a RESULT there if you have a resulting variable
      3. When taking in a variable, you must write “DATATYPE, INTENT(IN) :: variablename”
      4. When you intend to bring out a variable, you must write “DATATYPE, INTENT(OUT) :: variablename”
      5. Additionally, at the end of every function write, “END FUNCTION functioname)
2. Are there any rules about where the function has to be placed in your code file so that it can run?
   1. In Fortran, you need to place functions outside of the program code. Doing elsewise, by placing the functions inside, will basically break everything.
3. Does your language support recursive functions?
   1. In Fortran 90 and above, yes Fortran does support recursive functions. You simply need to add RECURSIVE to your function header, before the word FUNCTION. You also do not need to specify the data type when it comes to creating a recursive function.
4. Can functions in your language accept multiple parameters? Can they be of different data types?
   1. Functions can accept multiple parameters, as long as you make sure to write the line of code that accepts them (allows the compiler to read them IN). Judging from my brief experience, Fortran does not allow you to include different data types per say, mostly when it comes to accepting parameter variables.
5. Can functions in your language return multiple values at the same time? How is that implemented? If not, are there ways around that problem? What are they?
   1. Functions can return multiple values, by including them in RETURN but also making sure to either put that in RESULT or the INTENT(OUT) parts.
6. Is your language pass-by reference or value? Check your code against outside sources in case there is anything tricky going on (like in Perl).
   1. Majorily, Fortran is known to be a pass-by reference language, as it passes the reference of a value by its location. Using pointers, you can attempt to see how the compiler handles that. I tried to pass through two parameters, val, and ptr. I then attached the pointer to the value of val, and then printed them both separately.
7. Where are the arguments, parameters and local variables stored (value-on-stack, ref-to-heap-on stack) during execution?
   1. Despite a lengthy search online and through some deep thinking, I cannot come to an answer of where local, parameters, and arguments are specifically stored during execution in Fortran. Judging from an online source that I do not think is entirely accurate (though it’s the only thing finadable) says that Fortran doesn’t necessarily care? I do wonder if like other programming languages, Fortran places its local variables in the stack and global variables in the heap.
8. What are the scoping rules in your language (visibility and lifetime of variables before, during and after code blocks)?
   1. To begin with, this entirely depends on what flavor aka version of Fortran you're thinking about. Fortran-90 and before is an entirely complex matter but with Fortran-95, and especially Fortran-2008, scoping rules have greatly simplified. (I am refraining from discussing older Fortran versions, as I’m sure people nowadays are concerned only about the more recent flavors.) Modern Fortran flavors’ scoping rules are based on units. Is a variable by the same name in both a function and outside in the program? If yes, then Fortran will look at both and go; “These are two different variables, have a nice day”. Though, for Fortran-2008 and newer flavors, scope is defined by BLOCKS. Variables within a BLOCK will remain loyally local inside of it and be treated as such. Furthermore, technically these in-BLOCK variables only ‘exist’ within that BLOCK, think of it as some house it belongs to. Compared to the program-unit scoping rules of the past, we think of that as land or territory.
9. Are side-effects possible? Are there guard rails against side-effects?
   1. There isn’t a clear, exact answer online but one article from SpiceWorks, written by Chiradeep BasuMallick, states that data protection is an issue regarding Functions. The reasoning behind this side-effect is due to the fact that Fortran is usually a pass-by-reference language. My presumption on guard rails are to use extensions or since Fortran is *usually* a pass-by-reference, I wonder if there’s a way to have it by pass-by-value. Again, despite Fortran being an oldie but a goldie, there isn’t so much officially recorded.
10. Where are local variable values stored? (on the stack? On the heap?)
    1. My presumption is that local variables are stored in the stack in Fortran. Unfortunately, there are no concrete answers written on the web or in the official fortran website. My guess is based on some other programming languages and my brief experience as a CS student.
11. Are there any other aspects of functions in your language that aren't specifically asked about here, but that are important to know in order to write one? What are they? (e.g. dynamic vs static scope)
    1. When it comes to Fortran, there are actually ‘two different types of functions’. To begin with, Fortran has regular old functions that are created outside of the program blocks. They are called inside the program but need to anyway be introduced as variables in the program block. Functions are generally great for simplistic results and basic calculations.   
       However, subprograms are created inside the program block and behave a bit differently. Additionally, they may perform even more simplistic code such as basic calculations or taking in information from the user. When you need to refer to a subroutine, you would have to use CALL instead of simply writing out the name, which is unlike functions.

***SOURCES***

<https://fortran-lang.discourse.group/t/heap-stack-and-thread-safety/4944>

<https://www.spiceworks.com/tech/tech-general/articles/what-is-fortran/>

<https://web.chem.ox.ac.uk/fortran/subprograms.html>

<https://en.wikibooks.org/wiki/Fortran/strings>

<https://gist.github.com/ponderomotion/3833266>

<https://fortranwiki.org/fortran/show/m_strings>

<https://webpages.scu.edu/ftp/dsmolarski/ma61/notesforrec.html>

<https://www.tutorialspoint.com/fortran/fortran_pointers.htm>

<https://medium.com/@lucianoalmeida1/back-to-basics-pass-by-value-vs-pass-by-ref-96a591c88174>