Name: \_\_\_\_\_\_Joe Leveille\_\_\_\_\_\_

Student ID: \_\_\_\_\_\_\_\_\_\_\_\_3955537\_\_\_\_\_\_\_\_\_\_\_

HC: I have neither given or received, nor have I tolerated others’ use of unauthorized aid.

0) Commenting tools are available in \*\*\*\*\*\*\*every\*\*\*\*\*\*\* language (besides TI Basic, apparently).

True or False?

1) Discuss why we care about the difference between compilation and interpretation….

(at least 2 categories of difference, please)

Compiled programs generally run faster than interpreted programs due to a compiler making a program into machine code in a separate step before the program is run. Interpreters attempt to perform both steps in one, making for more lengthy run times.

Interpreted programs will work on any computer with the necessary interpreter, as opposed to compiled programs that must be compiled on a computer of the correct architecture. (i.e. a program compiled on a Windows machine would not agree with a Mac OS computer)

2) Explain the minimum two cases that we need for recursion:

(why is that a minimum - not a maximum??)

A recursive function needs a(n):

1. recursive case in which it calls itself
2. end case that breaks the recursion loop

because:

1. without a recursive case, there is no looping through the same function happening.
2. without an end case, the loop will be infinite, which will not allow the program to run all the way through.

Without these pieces together, the program will not properly perform a recursive task. There could be more cases to check for in more complicated recursive applications, but without the above two cases, there could not be any recursion.

3) Describe two major innovations from Lisp….

Recursion – calling a function from within itself to accomplish a repetitive task

Garbage Collection – Automatic memory managing (so that all the memory doesn’t get filled with unnecessary data from previous instances of a program

4) What does

(car (cdr (cdr '(a b c d))))

Get evaluated as? \_\_\_c\_\_\_\_

….. And briefly describe (in lisp terms) what’s going on:

The car command can be seen as “look at the first item in the given list”, and the cdr command can be seen as “look at the rest of the list without the first item”. As a result…….

First, the list (a b c d) gets shortened by the innermost cdr command to (b c d). Then the second cdr command shortens (b c d) to (c d), followed by the car command returning the leftmost element in (c d), which is c.

5) Is Python a “pure” scripting language, why or why not?

I would not consider Python a pure scripting language because it can still be used very similarly to lower level programming languages like C if the programmer does not decide to use built-in tools or easily-available libraries. This language could certainly be used at a high level of abstraction using the numerous libraries out there, but that does not remove the option of working from scratch, so it could be used as a non-scripting language.

6) What must every programming language ultimately result in/become….? (Otherwise, well, it isn’t useful.)

machine code

7) List and discuss at least 2 factors that make a language “succeed” (get used) or “fail” (get thrown away/be ignored?)

1. Usability – In order for a language to get used, it must be readable, learnable and have some sort of documentation for new users to reference, otherwise it would have no chance of attracting programmers to it.
2. Purpose – A language must have some application or aspect where it excels over other languages, whether that is faster runtime, more available libraries, or helping to fulfill a need in a niche area of programming. If a language had no qualities that were better than it’s competitors’, then it would quickly get tossed to the curb (excluding old applications that nobody wants to update).

8) Describe at least two big needs/area/challenges that the scripting languages tend to be the solution for… as in, where are they found?

Scripting languages are often used to:

1. Make a program portable – These languages are usually interpreted instead of compiled, so they can run just as easily on any computer on which they are run.
2. Automating individual tasks – Scripting languages are more high-level, so they try to do away with the specifics about how a task is handled every step of the way. Instead, languages like Python have access to many functions that take care of tasks that would take many more lines of code in a non-scripting language.

9) (doubled…) Besides python, what is one \*other\* example of a scripting language. What is it used for?

JavaScript is a scripting language that is used for web apps because of it’s numerous tools and libraries that help automate mundane code for seemingly simple tasks. (ex: React.js for changing HTML on the fly)

10) (also a double) Discuss the tradeoffs we make between a compiled and an interpreted language? What **<= I’m going to assume that was meant to be a follow up question, but proceed by ignoring it**

Compiled languages create separate files with machine code (aka executables), which are specific to the type of computer on which it gets compiled, whereas interpreted languages get compiled at runtime, essentially in one step.

Compiled languages run faster because they are compiled before running.

Compiled languages are more annoying to test small changes because they have both those steps needed to compile and run a new version of a program.

Compiled languages tend to be lower level languages, which do not have as many built-in tools as higher level, interpreted languages.

Overall, compiled languages are faster for the computer and interpreted languages are quicker for the programmer.

~~11) What must every programming language ultimately result in/become….?  (Otherwise, well, it isn’t useful.) (THIS IS A REPEAT FROM ABOVE - QUESTION 6)~~

12) (also worth double) Discuss - *with some detail, please* - at least one example of where a functional language gets used --- and how the various attributes, as in the criteria we use to describe/categorize/label/discuss a language - are relevant to that use.

Lisp is used in AutoCAD’s command line because:

* lisp allows arbitrary sizes of lists, allowing tons of pieces/models to be used together.
* functional languages are known for reliability over time. (hence the long-time usage)
* lisp is interpreted
* lisp can automate smaller tasks by creating functions

13) Explain the difference between imperative and declarative languages.

Imperative languages tend to be lower level, like C, which require the programmer to explicitly tell the computer to perform each small piece of a task step-by-step. These languages generally deal with the specifics on how a job gets completed. (ex: manual memory allocation, not many built-in functions)

Declarative languages, like Python, instead simplify the programmer’s job by including plenty of built-in functions that allow one command to take care of the specific low-level steps automatically. From a programmer’s perspective, these languages deal more with what a program is doing overall, since the high level approach removes the programmer from the step-by-step that is going on underneath the functions.

14) Which of the languages you’ve used to date (not just in class) do you like the most. Why… in PL terms to the extent possible.

Python is my favorite language because it is easily readable, the syntax is intuitive & simple, it is easily applicable to jobs that I may have in the future, and there are a wide variety of available libraries that contain massive amounts of useful tools. It is a high level, declarative, interpreted language with plenty of current users and online documentation, all of which make it simple to use (and learn) for the programmer.

~~15) List the two ways we discuss typing. (as in, the spectra? axes? set of labels? we use)~~

Pointless language humor:

