Q1

9 Points

Define the von Neumann’s architecture. Include both how a von Neumann machine is wired and the process used to execute programs.

The von Neumann was the idea of letting a computer store memory in order to execute instructions. It was theorized in order to reduce the repetition of telling a computer instructions repetitively, especially if it was already mentioned before. It is wired permanently with a given set of operations and storing it into the machine's memory, in which the instructions would be executed in the order they were given.

Q2

9 Points

Define abstraction and describe why it's a necessary feature of high-level languages.

Abstraction is the method of hiding away underlying functionality in order to simplify the overall purpose of a machine or operation. It is a necessary feature of high-level languages because it allows programmers to focus on the logical semantics of their program instead of the tedious memory management.

Q3

9 Points

List, in alphabetical order, the computational paradigms discussed *in lecture*.

Imperative

Functional

Logic

Scripting

Object Oriented

Q4

9 Points

How are compilers and interpreters similar? How are they different?

Compilers and interpreters are similar because they are both translators that convert code from one language to another. They are different because compilers translate programs into machine language before they are executed while interpreters translate programs into machine language at runtime. This results in compilers generally leading to better performance while interpreters are slightly slower (not too noticable given the hardware in present-day) since the interpreter reads statements in that language about one line at a time, and executes them as it reads them.

Q5
9 Points

What two language design criteria focus areas would you prioritize when designing a programming language? Why?

Two language design criteria focus areas I would prioritize when designing a programming language would be programmer efficiency and security since I would want to ensure that programmers have an easy time with programming in the language. I would also want to ensure that someone's programming mistakes don't destroy their machine, even if it means limiting the powerful-ness of the language.

Q6
9 Points

Discuss two or more examples of efficiency in the Java programming language. Specify if your example is computational efficiency or programmer efficiency.

Two examples of efficiency in the Java programming language is its mixture between using compiling and interpreting since it is a safe mixture between providing decent troubleshooting for syntax errors and still being incredibly speedy, which is computational efficiency. Another example would be the block structure that Java uses, which would be the combination of curly braces and the indentation since it speeds up the programmer's efficiency since it is easier to see chunks of code like loops and method definitions.

Q7

9 Points

Discuss two or more examples of inefficiency in the Java programming language. Specify if your example is computational efficiency or programmer efficiency.

One example of inefficiency in the Java programming language is having the combination of interpreter and compiler. If it just used compiling without interpreting, it would run much faster since the data types would be checked already, but the error message would be almost useless to the programmer. This would be an example of poor computational efficiency. Although the difference is small, it is bigger in the scheme of compiler versus interpreter. A programmer efficiency problem could be a programmer who decides not to indent in Java since it technically is not required, but it is more of a unwritten writing in Java rule. This would make it much slower to read Java code and collaborate with others.

Q8

9 Points

Discuss two or more examples of regularity in the Java programming language.

Two examples of regularity in the Java programming language are having the block code format of indenting and using curly braces, which is uniformity. In addition, using .equals() will always compare for equality across different data types, which is generality so there are not special cases.

Q9

9 Points

Discuss two or more examples of irregularity in the Java programming language.

An example of irregularity in the Java programming language would be the lack of uniformity in Java where a method function definition uses curly braces, but when it is called, it uses semi-colons. Another example would be the irregularity of primitive data types and reference types in Java which makes it very confusing and requires the programmer to know how to use parsing in detail.

Q10

10 Points

Languages such as Lisp and Python allow variables names to be used without declarations, while C, Java, and Ada require all variables to be declared before use. Should a language require the declaration of variables? Justify your answer from the perspective of both programmer efficiency and security.

Languages should require the declaration of variables since it would be more uniform in the context of older and outdated languages also requiring data types be declared when they are instantiated. From the perspective of programmer efficiency, it would just make it easier since programmers from C, Java, or Ada, or any preceding languages would be used to doing so. The only reason why someone would not expect to declare their data type is if they learned how to program using one of the more simple languages such as Python. In terms of security, it would be better for a program to automate it all since the programmer would not be able to force one data type to another, but this does remove the flexibility. I feel like it is essential to be able to convert stuff from arrays into just normal data types, but I think it's also very important to be able to declare things like this in order for those functionalities to work and be consistent with the idea of uniformity.

Q11
9 Points

List, in alphabetical order, the two types of programming language abstractions and the three levels of programming language abstractions

Abstraction types:

Data abstraction

Control abstraction

Abstraction levels:

Basic abstractions

Structured abstractions

Unit abstractions

Assignment 1

● Graded

Select each question to review feedback and grading details.

Student
Rin Pereira

Total Points

86.5 / 100 pts

Question 1 (no title)	**9** / 9 pts
Question 2 (no title)	**7.5** / 9 pts
Question 3 (no title)	**9** / 9 pts
Question 4 (no title)	**4.5** / 9 pts
Question 5 (no title)	**6** / 9 pts
Question 6 (no title)	**9** / 9 pts
Question 7 (no title)	**9** / 9 pts
Question 8 (no title)	**9** / 9 pts
Question 9 (no title)	**9** / 9 pts
Question 10 (no title)	**5.5** / 10 pts
Question 11 (no title)	**9** / 9 pts