1. Discuss what factors influence the design of programming languages.

Machine architecture and software development methodologies.

2. From the perspective of the history of programming languages, discuss the most important factor that makes a language successful.

Historically, machine architecture influences the most. E.g., von Neumann computers make imperative languages most dominant.

3. What arguments can you make for the idea of a single language for all programming domains?

Some arguments for having a single language for all programming domains are: It would dramatically cut the costs of programming training and compiler purchase and maintenance; it would simplify programmer recruiting and justify the development of numerous language dependent software development aids.

4. What arguments can you make against the idea of a single language for all programming domains?

Some arguments against having a single language for all programming domains are: The language would necessarily be huge and complex; compilers would be expensive and costly to maintain; the language would probably not be very good for any programming domain, either in compiler efficiency or in the efficiency of the code it generated. More importantly, it would not be easy to use, because regardless of the application area, the language would include many unnecessary and confusing features and constructs (those meant for other application areas). Different users would learn different subsets, making maintenance difficult.

5. What are the most common language evaluation criteria?

Readability, Writability, Reliability and Cost.

6. Assuming a language allows a function to return results of some types but not all types (like C where arrays cannot be returned). What principle is violated?

Orthogonality Principle: A relatively small set of primitive constructs can be combined in a relatively small number of ways, which ensures that every possible combination is legal.

7. What are the advantages of the principle mentioned in Question 6? i.e. what language criteria does this principle support? Could there be disadvantages if it was overused in language design?

The orthogonality principle supports readability and writebility. It allows few constructs, a small number of primitives, and a small set of rules for combining them, which makes the program easier to be read and written.

One possible disadvantage of overusing orthogonality principle is wordiness. In some languages, a great deal of text is required for even simple complete programs. For example, COBOL is a very wordy language. In Ada, programs require a lot of duplication of declarations. Wordiness is usually considered a disadvantage, because it slows program creation, takes more file space for the source programs, and can cause programs to be more difficult to read.

8. Java uses a right brace to mark the end of all compound statements. What are the arguments for and against this design?

The argument for using the right brace to close all compounds is simplicity—a right brace always terminates a compound. The argument against it is that when you see a right brace in a program, the location of its matching left brace is not always obvious, in part because all multiple-statement control constructs end with a right brace.

9. Explain the different aspects of the cost of a programming language.

- Training programmers to use language
- Writing programs “Writability”
- Compiling programs
- Executing programs
- Language implementation system “Free compilers is the key, success of Java”
- Reliability, does the software fail?
- Maintaining programs: Maintenance costs can be as high as two to four times as much as development costs.
- Portability “standardization of the language”
- Generality (the applicability to a wide range of applications)

10. What are the arguments for writing efficient programs even though hardware is relatively inexpensive?

One of the main arguments is that regardless of the cost of hardware, it is not free. Why write a program that executes slower than is necessary. Furthermore, the difference between a well-written efficient program and one that is poorly written can be a factor of two or three. In many other fields of endeavor, the difference between a good job and a poor job may be 10 or 20 percent. In programming, the difference is much greater.

11. Many contemporary languages allow two kinds of comments: one in which delimiters are used on both ends (multiple-line comments), and one in which a delimiter marks only the beginning of the comment (one line comments). Discuss the advantages and disadvantages of each of these with respect to the language evaluation criteria discussed in lectures.

The main disadvantage of using paired delimiters for comments is that it results in diminished reliability. It is easy to inadvertently leave off the final delimiter, which extends the comment to the end of the next comment, effectively removing code from the program. The advantage of paired delimiters is that you can comment out areas of a program. The disadvantage of using only beginning delimiters is that they must be repeated on every line of a block of comments. This can be tedious and therefore error-prone. The advantage is that you cannot make the mistake of forgetting the closing delimiter.