

CS 373 Term Project  
Spring 2008

Choose a programming language from among Ada, C, C++, C#, COBOL, Scheme, Fortran, Lisp, Java, Prolog, Visual Basic, Perl, HTML, PHP, or any other of your choice. Answer the following questions. Write appropriate code to support your answers.

1. Classify your language (imperative, functional, logic, OO, ?).
2. Is the scoping static or dynamic?
3. Is type checking static, dynamic, neither, or some combination? What are the built-in types? Which are primitive?
4. When are two types equivalent? Are there exceptions? Be sure to include user defined types in your consideration.
5. What parameter modes are available? Comment on whether or not these modes are adequate for good programming practice.
6. What kinds of parameters can be passed, i.e., can functions or procedures be passed? Comment on any deficiencies and give suggestions for addressing those deficiencies.
7. If there is a case (switch) statement, what happens when an unspecified choice is evaluated for the expression controlling the case statement? What kinds of expressions are permissible for controlling the case statement and why?
8. How are cartesian products supported? Are there variant records? If so, what happens if you ask to print a value that a given record should not have? If not, is there some way to compensate? Comment on the desirability of variant records and suggest how they might be implemented. Can an array contain entries of different types or must they all be of the same type? How much about the implementation does a programmer need to know in order to use arrays and records efficiently?
9. If there are **for** loops, does the language permit jumping into the loop? Modifying the loop control variable inside the loop? Jumping out of the loop?
10. What kinds of iteration are supported? For example, **while**, **repeat-until**, **do**, etc. Are they all equivalent? Explain how they differ and how they are alike by writing code in each of the syntactic possibilities for a single problem with an iterative solution. Is it possible to exit from any of these iterative constructs before the natural end is reached? If so, how and where does the control resume? If not, tell how such an exit might be added to the language.
11. Does your language support exceptions? If so, explain how they work. Does your language support assertions? If so, explain how they work.
12. How are dynamic variables ( pointers ) handled? Are there dynamic arrays? If so, how are they implemented? Is this a good idea? Why or why not?
13. What happens if you write  $A := B$  (or  $A = B$ ) when both A and B are arrays? when both A and B are records? when both A and B are of the same user defined type? Note: the answer may depend on operator overloading depending on the language. Give reasons why it might be a good idea to allow variables of user defined types to be assigned to each other

and reasons why it might not be a good idea.

14. Find 3 classes of errors which your translator alerts you to and 3 which it does not.
15. In what ways are user-defined types treated differently from built in types? In what ways are they treated the same?
16. Are there limits for arithmetic computations (such as a minimal or maximal integer)? What happens if you ask to perform computations that cause overflow? Explain what kinds of numbers are built in, what limitations they have with regard to precision, and how they are stored.
17. Find 3 cases of automatic type coercion, if possible. What explicit type conversion capabilities are built in? Is explicit type conversion required? What can be converted to what and what cannot?
18. Is there type inheritance?
19. Is polymorphism possible? If so, to what extent?
20. Is there support for exception handling? How does it work?
21. Are there any built-in features to support concurrency? If so, give some examples.
22. To what extent does your language support information hiding? To what extent does your language support encapsulation? Can you nest blocks within blocks (classes within classes, methods within methods, classes within methods, loops within loops) ? Give examples.
23. To what extent does your language support genericity? Give an example of a case where the generics support multiple use of the same code. Give an example for which generic support does not exist.
24. To what extent does your language support hierarchical construction of programs from reusable parts? What about separate compilation of the parts?
25. Does your language provide any support for defensive programming? (Assert)?
26. Give 3 characteristics of your language that make it easy to compile. Give 3 that make it difficult to compile. Explain.
27. Analyze the syntax of your chosen language by finding some particularly useful constructs and styles and by pointing out some limiting aspects of the constructs.
28. Does your language have formal syntax? If so, in what form? Does your language have formal semantics? Where does a compiler constructor go for semantics?
29. Are there both interpreters and compilers for your language? How can each be used? Is your language all-purpose? The answer is probably “yes.” With that in mind, discuss how hard or easy a variety of tasks would be to do in your language. For example, operating system code, graphics applications, database management, system level code, application level code, large programs, small programs, mathematical applications such as the space program, networking applications, others of your choice.
30. Is your language all-purpose? The answer is probably “yes.” With that in mind, discuss how hard or easy a variety of tasks would be to do in your language. For example, operating system code, graphics applications, database management, system level code, application level code, large programs, small programs, mathematical applications such as the space program, networking applications, others of your choice.

**Write a brief history** of your chosen language. What were the goals of the language designers? What motivated the introduction of the language? What languages are in its ancestry?

**Write documentation** which tells how to use your chosen language on the equipment at Denison. Your audience is someone who is an experienced programmer, but not in your language or not in the equipment on campus. You may assume that the reader can access a language manual either in paper or online. You should list keywords with an explanation and an example to illustrate each. You should describe the environment in which programming in your language must take place. This includes identifying how to edit, compile, and save programs, as well as what, if any, debugging support there is. Depending on the language, there may be libraries of available programs. You should explain how to find them and use them. Part of your explanations may tell the reader how to use the help feature if one is available. You should provide some examples for the reader to try out to get started.

Most of the languages you have been studying claim to be all-purpose languages, yet each has features different from the others. **Choose at least 3 features unique to your language** and explain how those features can be used advantageously by a programmer to perform some task in your language more efficiently than it could be done in another language without those features.

Be prepared to give a talk and a **demonstration** which illustrate how your language facilitates the implementation of solutions to problems.

Note: Your documentation will be given to other(s) in the class. They will critique it after trying to use the language based on your instructions.