Lista de Exercícios de Linguagens de Programação — Capítulo 1.
Primeiro Semestre de 2000. Dept. de Computação – UFSCar.
José de Oliveira Guimarães.

1. Dê um exemplo de ambigüidade na definição de uma linguagem de programação. Explique onde está a ambigüidade.

2. Defina linguagem de propósito geral.

3. Cite três características de uma boa linguagem de programação.

4. Defina ortogonalidade. Cite um exemplo.

5. Em que casos ortogonalidade pode não ser uma boa característica ? Cite um exemplo.

6. Qual o melhor argumento a favor da ortogonalidade em linguagens de programação ?

7. Cite quatro desvantagens de se ter uma linguagem muito grande.

8. Cite uma possível desvantagem de uma linguagem muito pequena.

9. Ao escolher uma linguagem de programação para ser utilizada em um projeto alguns fatores independentes da linguagem devem ser considerados. Cite estes fatores.

10. Defina portabilidade.

11. Portabilidade refere-se ao código fonte, objeto ou executável ?

12. Considere que uma linguagem L foi mal especificada e possui algumas construções ambíguas. Que dificuldades poderão surgir ao transportar um programa de um compilador para outro ? Transportar significa compilar o código fonte no outro compilador.

13. Compiladores tomam um código fonte e produzem um código em uma linguagem destino. Que linguagens são mais freqüentemente utilizadas como destino ?

14. Qual a diferença fundamental entre código compilado e interpretado ?

15. Faça um exemplo de compilação/ligação com dois arquivos fonte ".c" que são compilados para dois arquivos ".obj" e que resultam em um arquivo executável. Cada arquivo ".c" define uma função que é chamada no outro arquivo. Faça os retângulos como está na apostila mostrando como ficará o executável. Não é necessário decorar o formato dos arquivos ".obj".

16. Defina sistema de tempo de execução e cite cinco de suas atribuições.

17. Cite quatro vantagens de interpretadores sobre compiladores.

18. Explique porque a seqüência Edite-Compile para bytecodes-Interprete utilizada por interpretadores é mais rápida do que a seqüência Edite-Compile-Ligue-Execute utilizada por compiladores.

19. Escreva a hipótese (chamada erroneamente de tese) de Church-Turing.

20. Suponha que alguém encontre um algoritmo que não possa ser implementado por nenhuma linguagem de programação. Neste caso, a tese de Church-Turing ainda será válida ?

21. Prove que não é possível existir um algoritmo que toma o texto de um outro algoritmo como parâmetro e diz se ele pára ou não.