01. [Sebesta, 2000] Defina precedência de operadores e associatividade de operadores.

02. [Sebesta, 2000] Defina efeito colateral funcional.

03. [Sebesta, 2000] O que é uma coerção?

04. [Sebesta, 2000] O que é uma expressão condicional?

05. [Sebesta, 2000] O que é um operador sobrecarregado?

06. [Sebesta, 2000] Defina conversões de estreitamento e de alargamento.

07. [Sebesta, 2000] O que é uma expressão de modo-misto?

08. [Sebesta, 2000] Como a ordem de avaliação de operandos interage com efeitos colaterais funcionais?

09. [Sebesta, 2000] O que é avaliação curto-circuito?

10. [Sebesta, 2000] Cite uma linguagem que sempre faça avaliação curto-circuito de expressões booleanas. Cite uma que nunca a faça. Cite uma em que sempre seja permitido ao programador escolher.

11. [Sebesta, 2000] Como o C suporta expressões relacionais e booleanas?

12. [Sebesta, 2000] Qual é o propósito de um operador de atribuição composto?

13. [Sebesta, 2000] Qual é a associatividade dos operadores aritméticos unários do C?

14. [Sebesta, 2000] Qual é uma das possíveis desvantagens de tratar o operador de atribuição como se ele fosse um operador aritmético?

15. [Sebesta, 2000] Quais atribuições de modo misto são permitidas na Ada?

16. [Sebesta, 2000] Quais atribuições de modo misto são permitidas no Java?

17. [Sebesta, 2000] Quando você poderia querer que o compilador ignorasse diferenças de tipos em uma expressão?

18. [Sebesta, 2000] Apresente seus próprios argumentos contrários e favoráveis a permitir expressões aritméticas de modo misto.

19. [Sebesta, 2000] Você acha que a eliminação de operadores sobrecarregados de sua linguagem favorita seria benéfica? Por que sim e por que não?

20. [Sebesta, 2000] Seria uma boa ideia eliminar todas as regras de precedência de operadores e exigir parênteses para mostrar a precedência desejada nas expressões? Por que sim e por que não?

21. [Sebesta, 2000] As operações de atribuição do C (por exemplo, +=) devem ser incluídas em outras linguagens? Por que sim e por que não?

22. [Sebesta, 2000] As formas de atribuição de único operando do C (por exemplo, `++count`) devem ser incluídas em outras linguagens? Por que sim e por que não?

23. [Sebesta, 2000] Descreva uma situação em que o operador de adição em uma linguagem de programação não seria comutativa.

24. [Sebesta, 2000] Descreva uma situação em que o operador de adição em uma linguagem de programação não seria associativo.

25. [Sebesta, 2000] Escreva um segmento de programa Pascal, usando uma construção `while` para procurar um *array* de números inteiros para um número inteiro particular, que funcionaria mesmo que uma avaliação curto-circuito de expressões booleanas não fosse feita. Suponha as seguintes regras de associatividade e precedência para expressões:

Precedência:	Mais alta	*, /, not
		+, -, &, mod
		- (unário)
		=, /=, <, <=, >=, >
		and
	Mais baixa	or, xor
Associatividade	Da esquerda para a direita	

26. [Sebesta, 2000] Mostre a ordem de avaliação das seguintes expressões colocando entre parênteses todas as sub-expressões e colocando em sobrescrito no parêntese direito para indicar a ordem. Por exemplo, para a expressão `a + b * c + d` a ordem de avaliação seria representada como $((a + (b * c)^{1})^{2} + d)^{3}$.

a) `a * b - 1 + c`

b) `a * (b - 1) / c mod d`

c) `(a - b) / c & (d * e / a - 3)`

d) `-a or c = d and e`

e) `a > b xor c or d <= 17`

f) `-a + b`

27. [Sebesta, 2000] Mostre a ordem de avaliação das expressões do exercício 26, supondo que não haja quaisquer regras de precedência e que todos os operadores associem-se da direita para a esquerda.

28. [Sebesta, 2000] Escreva um descrição BNF das regras de precedência e de associatividade definidas para as expressões do exercício 26. Suponha que os únicos operandos sejam os nomes a, b, c, d e e.

29. [Sebesta, 2000] Usando a gramática do exercício 28, desenhe árvores de análise das expressões do exercício 26.

30. [Sebesta, 2000] Admitindo que a função `FUN` seja definida como

```
function FUN(var K : integer) : integer;
begin
  K := K + 4;
  FUN := 3 * K – 1;
end;
```

Suponha que `FUN` seja usada em um programa da seguinte maneira:

```
...
I := 10;
SOMA1 := (I / 2) + FUN(I);
J := 10;
SOMA2 := FUN(J) + (J / 2);
```

Quais são os valores de `SOMA1` e `SOMA2`?

a) os operandos nas expressões forem avaliados da esquerda para a direita?

b) os operandos nas expressões forem avaliados da direita para a esquerda?

31. [Sebesta, 2000] Admitindo que a função C `fun` seja definida como

```
int fun(int *k) {
  *k += 4;
  return 3 * (*k) – 1;
}
```

Supondo que `fun` seja usada em um programa da seguinte maneira:

```
void main() {
  int i = 10, j = 10, soma1, soma2;
  soma1 = (i / 2) + fun(&i);
  soma2 = fun(&j) + (j / 2);
}
```

Determine os valores de soma1 e soma2 rodando o programa em um computador. Explique os resultados.

32. [Sebesta, 2000] Qual é seu principal argumento contrário (ou favorável) às regras de precedência de operadores da APL?

33. [Sebesta, 2000] Para alguma linguagem de sua escolha, componha uma lista de símbolos de operadores que possa ser usada para eliminar toda sobrecarga de operador.

34. [Sebesta, 2000] Determine se as conversões de tipo explícitas de estreitamento em duas linguagens que você conhece apresentam mensagens de erro quando um valor convertido perde sua utilidade.

35. [Sebesta, 2000] Deve-se permitir que um compilador de otimização para C ou para C++ mude a ordem de sub-expressões em uma expressão booleana? Por que sim e por que não?

36. [Sebesta, 2000] Responda o exercício 35 em relação a Ada.

37. [Sebesta, 2000] Considere o seguinte programa C:

```
int fun(int *i) {
  *i += 5;
  return 4;
}

void main() {
  int x = 3;
  x = x + fun(&x);
}
```

Qual é o valor de `x` depois da instrução de atribuição em `main`, supondo que:

a) os operandos são avaliados da esquerda para a direita?

b) os operandos sã avaliados da direita para a esquerda?

38. [Sebesta, 2000] Escreva um programa de teste em sua linguagem favorita que determine e apresente como resultado a precedência e a associatividade de seus operadores aritméticos e booleanos.

39. [Sebesta, 2000] Admitindo que a função C `fun` seja definida como

```
int fun(int *k) {
  *k += 6;
  return 2 * (*k) – 1;
}
```

Supondo que `fun` seja usada em um programa da seguinte maneira:

```
void main() {
  int i = 20, j = 20, soma1, soma2;
  soma1 = (i / 2) + fun(&i);
  soma2 = fun(&j) + (j / 2);
}
```

Quais são os valores de `soma1` e `soma2` depois da instrução de atribuição em `main`, supondo que:

a) os operandos são avaliados da esquerda para a direita?

b) os operandos sã avaliados da direita para a esquerda?

40. Desenvolver um programa na sua linguagem de programação favorita, que apresente a ordem de avaliação dos operandos utilizada pela linguagem de programação selecionada. Explicar o que está acontecendo no código desenvolvido, e se há ou não a ocorrência de "efeito colateral" na linguagem.