01. [Sebesta, 2000] Defina exceção, manipulador de exceções, levantamento de uma exceção, desativação de uma exceção e exceção incorporada.

02. [Sebesta, 2000] Quais são as questões de projeto relativas à manipulação de exceções?

03. [Sebesta, 2000] O que se quer dizer por uma exceção estar vinculada a um manipulador de exceções?

04. [Sebesta, 2000] Qual é o problema com a vinculação de exceções a manipuladores da PL/I?

05. [Sebesta, 2000] Quais são os quadros possíveis para exceções em Ada?

06. [Sebesta, 2000] Onde as exceções não-manipuladas são propagadas em Ada se elas forem geradas em um subprograma? Em um bloco? Em um corpo de pacote? Em uma tarefa?

07. [Sebesta, 2000] Onde a execução prossegue depois que uma exceção é manipulada em Ada?

08. [Sebesta, 2000] Como uma exceção pode ser explicitamente gerada em Ada?

09. [Sebesta, 2000] Como uma exceção definida pelo usuário é definida em Ada?

10. [Sebesta, 2000] Como uma exceção pode ser suprimida em Ada?

11. [Sebesta, 2000] Qual é o nome de todos os manipuladores de exceção do C++?

12. [Sebesta, 2000] Como as exceções podem ser explicitamente geradas em C++?

13. [Sebesta, 2000] Como as exceções são vinculadas a manipuladores em C++?

14. [Sebesta, 2000] Como um manipulador de exceções pode ser escrito de maneira que manipule qualquer exceção?

15. [Sebesta, 2000] Para onde vai o controle de execução quando um manipulador de exceções do C++ encerra sua execução?

16. [Sebesta, 2000] O C++ inclui exceções incorporadas?

17. [Sebesta, 2000] Qual é a classe-raiz de todas as classes de exceção Java?

18. [Sebesta, 2000] Qual é a classe-pai da maioria das classes de exceção Java definidas pelo usuário?

19. [Sebesta, 2000] Como um manipulador de exceções pode ser escrito em Java de maneira que manipule qualquer exceção.

20. [Sebesta, 2000] Qual é a diferença entre uma especificação `throw` C++ e uma classe `throws` Java?

21. [Sebesta, 2000] Qual é a diferença entre exceções verificadas e não-verificadas em Java?

22. [Sebesta, 2000] Como se pode desativar uma exceção Java?

23. [Sebesta, 2000] Qual é o propósito da cláusula Java `finally`?

24. [Sebesta, 2000] Quais erros ou condições que, se for o caso, programas Pascal, em tempo de execução, podem detectar ou manipular?

25. [Sebesta, 2000] Em livros didáticos sobre as linguagens de programação PL/I e Ada, pesquise os conjuntos respectivos de exceções incorporadas. Faça uma avaliação comparativa das duas, considerando sua inteireza como sua flexibilidade.

26. [Sebesta, 2000] Escreva um segmento de código Ada que recupere uma chamada a um procedimento, `tape_read`, que leia entradas de uma unidade de fita e possa gerar a exceção `tape_read`.

27. [Sebesta, 2000] Em *The Programming Language Ada Reference Manual* (Goos e Hartmanis, 1983), determine como são manipuladas as exceções que se desenvolvem durante o *rendezvous*.

28. [Sebesta, 2000] Em um livro didático sobre o COBOL, determine como é feita a manipulação de exceções em seus programas.

29. [Sebesta, 2000] Em linguagens sem facilidades de manipulação de exceções, é comum que a maioria dos subprogramas inclua um parâmetro "error", que pode ser fixado em algum valor que representa "OK" ou algum outro valor que representa "erro de procedimento". Quais vantagens uma facilidade de manipulação de exceções linguísticas como a Ada tem sobre esse método?

30. [Sebesta, 2000] Em uma linguagem sem facilidades de manipulação de exceções, poderíamos enviar um procedimento de manipulação de erros como um parâmetro a cada procedimento que possa detectar erros que devem ser manipulados. Quais desvantagens há nesse método?

31. [Sebesta, 2000] Compare os métodos sugeridos nos Problemas 29 e 30. Qual deles você acha que é melhor e por quê?

32. [Sebesta, 2000] Compare as facilidades de manipulação de exceções do C++ com as da Ada. Qual projeto, em sua opinião, é o mais flexível? Qual deles possibilita escrever programas mais confiáveis?

33. [Sebesta, 2000] Suponhamos que você esteja escrevendo um procedimento, na linguagem de programação de sua preferência, com três métodos alternativos para cumprir suas exigências. Escreva uma versão esquemática deste procedimento de maneira que, se a primeira alternativa gerar alguma exceção, a segunda seja experimentada, e se também gerar uma exceção, a terceira seja executada. Escreva o código como se os três métodos fossem procedimentos chamados ALT1, ALT2 e ALT3.

34. [Sebesta, 2000] Escreva um programa Ada que introduza uma lista de valores inteiros na faixa de -100 a 100 pelo teclado e compute a soma dos quadrados dos valores introduzidos. Este programa deve usar manipulação de exceções para assegurar que os valores introduzidos estejam dentro da faixa e sejam números inteiros legais, para manipular o erro da soma dos quadrados que tornar-se maior do que aquilo que uma variável `INTEGER` padrão pode armazenar, e detectar o fim de arquivo e usá-lo para acarretar a saída do resultado. No caso de *overflow* da soma, uma mensagem de erro deve ser impressa e o programa finalizado.

35. [Sebesta, 2000] Escreva um programa C++ para a especificação do Problema 34.

36. [Sebesta, 2000] Escreva um programa Java para a especificação do Problema 34.

37. [Sebesta, 2000] Escreva uma comparação detalhada das capacidades de manipulação de exceções do C++ e do Java.

38. [Sebesta, 2000] Considere o seguinte programa Java esquemático:

```
class Grande {
  int i;
  float f;
  void fun1() throws {int} {
    ...
    try {
      ...
      throw i;
      ...
      throw f;
      ...
    }
    catch(float) {...}
    ...
  }
}
```

```
class Pequena {
  int j;
  float g;
  void fun2() throws {float} {
    ...
    try {
      ...
      try {
        Grande.fun1();
        ...
        throw j;
        ...
        throw g;
        ...
      }
      catch(int) {...}
      ...
    }
    catch(float) {...}
  }
}
```

No caso das três instruções `throw`, onde está a exceção manipulada? Note que `fun1` é chamada de `fun2` na classe `Pequena`.

39. [Sierra, 2004] Dado o código a seguir,

```
1.  System.out.print("Start ");
2.  try {
3.  System.out.print("Hello world ")
4.  throw new FileNotFoundException();
5.  }
6.  System.out.print("Catch Here ");
7.  catch(EOFException e) {
8.  System.out.print("End of file exception ");
9.  }
10. catch(FileNotFoundException e) {
11. System.out.print("File not found ");
12. }
```

Sabendo-se que tanto `EOFException` quanto `FileNotFoundException` são subclasses de `IOException` e presumindo que esse bloco de código seja inserido em uma classe, que declaração estará mais perto da verdade com relação a esse código?

a) o código não será compilado.

b) saída do código: `Start Hello world File not found.`

c) saída do código: `Start Hello world End of file exception.`

d) saída do código: `Start Hello world Catch Here File not found.`

40. [Sierra, 2004] Dado o código a seguir,

```
1.  public class MyProgram {
2.  public static void main(String[] args) {
3.  try {
4.  System.out.print("Hello world ");
5.  }
6.  finally {
```

```
7.  System.out.println("Finally executing ");
8.  }
9.  }
10. }
```

Qual será o resultado?

a) nenhum. O programa não será compilado porque nenhuma exceção foi especificada.

b) nenhum. O programa não será compilado porque nenhuma cláusula `catch` foi especificada.

c) `Hello world`

d) `Hello world Finally executing`

41. [Sierra, 2004] Dado o código a seguir,

```
1.  import java.io.*;
2.  public class MyProgram {
3.  public static void main(String[] args) {
4.  FileOutputStream out = null;
5.  try {
6.  out = new FileOutputStream("test.txt");
7.  out.write(122);
8.  }
9.  catch(IOException io) {
10. System.out.println("IO Error.");
11. }
12. finally {
13. out.close();
14. }
15. }
16. }
```

E dado que todos os métodos da classe `FileOutputStream`, inclusive `close()` lançam uma `IOException`, qual dessas opções é verdadeira (selecione uma)?

a) esse programa será compilado com sucesso.

b) a compilação desse programa falhará devido a um erro na linha 4.

c) a compilação desse programa falhará devido a um erro na linha 6.

d) a compilação desse programa falhará devido a um erro na linha 9.

e) a compilação desse programa falhará devido a um erro na linha 13.

42. [Sierra, 2004] Dado o código a seguir,

```
1.  public class MyProgram {
2.  public static void throwit() {
3.  throw new RuntimeException();
4.  }
5.  public static void main(String[] args) {
6.  try {
7.  System.out.println("Hello world ");
8.  throwit();
9.  System.out.println("Done with try block ");
10. }
11. finally {
12. System.out.println("Finally executing ");
```

```
13. }
14. }
15. }
```

Que resposta está mais próxima de indicar o comportamento do programa?

a) o programa não será compilado.

b) o programa exibirá `Hello world`, depois exibirá que uma `RuntimeException` ocorreu e, ainda, mostrará `Done with try block` e depois `Finally executing`.

c) o programa exibirá `Hello world`, depois exibirá que uma `RuntimeException` ocorreu, e ainda, mostrará `Finally executing`.

d) o programa exibirá `Hello world`, depois exibirá `Finally executing`, e que uma `RuntimeException` ocorreu.

43. [Sierra, 2004] Dado o código a seguir,

```
1.  public class RTExcept {
2.  public static void throwit() {
3.  System.out.print("throwit ");
4.  throw new RuntimeException();
5.  }
6.  public static void main(String[] args) {
7.  try {
8.  System.out.print("hello ");
9.  throwit();
10. }
11. catch(Exception re) {
12. System.out.print("caught ");
13. }
14. finally {
15. System.out.print("finally ");
16. }
17. System.out.println("after ");
18. }
19. }
```

Qual será o resultado?

a) `hello throwit caught`

b) a compilação falhará

c) `hello throwit RuntimeException caught after`

d) `hello throwit RuntimeException`

e) `hello throwit caught finally after`

f) `hello throwit caught finally after RuntimeException`