01. [Sebesta, 2000] Quais são as quatro razões pelas quais implementar subprogramas em linguagens assemelhadas ao ALGOL é mais difícil do que implementar subprogramas em uma linguagem como o FORTRAN 77?

02. [Sebesta, 2000] Qual é a diferença entre um registro de ativação e uma instância do registro de ativação?

03. [Sebesta, 2000] Por que o endereço de retorno, o vínculo estático, o vínculo dinâmico e os parâmetros são colocados na parte inferior do registro de ativação?

04. [Sebesta, 2000] Quais são os dois passos para localizar uma variável não-local em uma linguagem de escopo estático, independentemente de qual método é usado?

05. [Sebesta, 2000] Defina encadeamento estático, *static_depth* (profundidade estática), *nesting_depth* (profundidade de aninhamento) e *chain_offset* (deslocamento de encadeamento).

06. [Sebesta, 2000] Explique como uma referência a uma variável não-local é localizada quando encadeamentos estáticos são usados.

07. [Sebesta, 2000] Quais são os dois problemas potenciais com o método de encadeamento estático?

08. [Sebesta, 2000] O que é um *display*?

09. [Sebesta, 2000] Explique como uma referência a uma variável não-local é localizada quando um *display* é usado.

10. [Sebesta, 2000] Como as referências a uma variável são representadas no método de encadeamento estático? Como elas são representadas no método de *display*?

11. [Sebesta, 2000] Quais mudanças no *display* são necessárias quando um subprograma é chamado (suponhamos que não haja nenhum parâmetro passado por nome e nenhum parâmetro que seja subprograma)?

12. [Sebesta, 2000] Compare a eficiência dos métodos de encadeamento estático e *display* para acessos locais, para acessos não-locais, para retornos e para globais.

13. [Sebesta, 2000] Explique os dois métodos para implementar blocos.

14. [Sebesta, 2000] Descreva o método de acesso profundo para implementar o escopo dinâmico.

15. [Sebesta, 2000] Descreva o método de acesso raso para implementar o escopo dinâmico.

16. [Sebesta, 2000] Quais são as duas diferenças entre o método de acesso profundo para acesso não-local em linguagens de escopo dinâmico e o de encadeamento estático para linguagens de escopo estático?

17. [Sebesta, 2000] Compare a eficiência do método de acesso profundo com o do método de acesso raso, tanto em termos de chamadas como de acessos não-locais.

18. [Sebesta, 2000] Descreva um método de implementar parâmetros que são subprogramas quando é usada a técnica de implementação de encadeamento estático.

19. [Sebesta, 2000] Descreva um método de implementar parâmetros que são subprogramas quando é usada a técnica de implementação de *display*.

20. [Sebesta, 2000] Em uma linguagem que permite parâmetros que são nomes de subprograma, a instância do registro correta de um pai estático é sempre a instância mais próxima do pai que está atualmente na pilha?

21. [Sebesta, 2000] Escreva um algoritmo para executar a manutenção do *display* exigida depois da entrada em um subprograma em uma linguagem que use escopo estático e que também permita nomes de subprograma como parâmetros.

22. [Sebesta, 2000] Escreva um algoritmo para executar a manutenção do display exigida depois da saída de um subprograma em uma linguagem que use escopo estático e que também permita nomes de subprogramas como parâmetros.

23. [Sebesta, 2000] Mostre a pilha com todas as instâncias do registro de ativação, incluindo encadeamentos estáticos e dinâmicos, quando a execução atingir a posição `(1)` no programa esquemático. Suponha que `BIGSUB` está no nível 1.

```
procedure BIGSUB;
  procedure A;
    procedure B;
    begin { B }
      .. (1) ..
    end; { B }
    procedure C;
    begin { C }
      ...
      B;
      ...
    end; { C }
  begin { A }
  ...
  C;
  ...
  end; { A }
begin { BIGSUB }
...
A;
...
end; { BIGSUB }
```

24. [Sebesta, 2000] Para o programa esquemático no Problema 23, mostre o *display* que estaria ativo na posição `(1)`, juntamente com as instâncias do registro de ativação na pilha.

25. [Sebesta, 2000] Mostre a pilha com todas as instâncias do registro de ativação, incluindo encadeamentos estáticos e dinâmicos, quando a execução atingir a posição `(1)` no programa esquemático seguinte. Suponha que `BIGSUB` esteja no nível 1.

```
procedure BIGSUB;
  procedure C; forward;
  procedure A (flag: boolean);
    procedure B;
    begin { B }
      ...
      A(false);
    end; { B }
  begin { A }
    if flag
      then B
      else C
    ...
  end; { A }
  procedure C;
    procedure D;
    begin { D }
      ..(1)..
    end; { D }
  begin { C }
    ...
    D;
  end; { C }
```

```
begin { BIGSUB }
  A(true);
  ...
end; { BIGSUB }
```

A sequência de chamada desse programa para que a execução atinja `D` é `BIGSUB` chama `A` que chama `B` que chama `A` que chama `C` que chama `D`.

26. [Sebesta, 2000] Em relação ao programa esquemático do Problema 25, mostre o *display* que estaria ativo na posição `(1)`, juntamente com as instâncias do registro de ativação na pilha.

27. [Sebesta, 2000] Não obstante as variáveis locais nos procedimentos Pascal serem alocadas dinamicamente no início de cada ativação, sob quais circunstâncias o valor de uma variável local em uma ativação particular poderia reter o valor da ativação anterior?

28. [Sebesta, 2000] Afirmamos neste capítulo, que quando variáveis não-locais são acessadas em uma linguagem de escopo dinâmico usando o encadeamento dinâmico, os nomes de variáveis devem ser armazenados nos registros de ativação com os valores. Se isso fosse realmente feito, todo acesso não-ocal exigiria uma sequência de custosas comparações de *strings* em nomes. Projete uma alternativa para essas comparações de *strings* mais rápida.

29. [Sebesta, 2000] O Pascal permite `goto` com alvos não-locais. Como essas instruções poderiam ser manipuladas se fossem usados encadeamentos estáticos para acesso a variáveis não-locais? *Dica*: considere a maneira pela qual a instância correta do registro de ativação do pai estático de um procedimento recém-ativado é encontrada.

30. [Sebesta, 2000] Repita o Problema 29, usando um *display* ao invés de encadeamentos estáticos.

31. [Sebesta, 2000] Como o mecanismo de *display* poderia ser descrito neste capítulo para tornar as variáveis locais acessíveis sem endereçamento indireto?

32. [Sebesta, 2000] O método de encadeamento estático poderia ser ligeiramente expandido usando-se dois vínculos estáticos em cada instância do registro de ativação, em que o segundo aponta para a instância avó estática do registro de ativação. Como isso afetaria o tempo necessário para a ligação de subprograma e de referências não-locais?