Resposta da questão 1

(a) As três regiões de memória definidas são:

- Área estática
- Pilha
- *Heap*

Cada uma corresponde a um tipo de alocação diferente.

A área estática é fixa, portanto ela não precisa de folga. Três variáveis no programa requerem alocação estáticas: as globais `a` e `b` e a local `c`. Assim, a área estática mede exatamente 3 unidades de memória. A área estática é convenientemente (ver abaixo) posicionada no topo da memória (posições `@23`, `@24`, `@25`).

A pilha e o *heap* dividem o restante da memória. Como o número de unidades de memória restantes é ímpar, a pilha é escolhida para ser maior, pois esse programa parece usar mais pilha do que *heap*. (Essas considerações são irrelevantes em casos reais, cujas medidas são da ordem de megabytes.)

A pilha cresce de cima para baixo (dos endereços maiores para os endereços menores). Dessa forma, "dá jogo": se o *heap* for pouco usado, a memória pode invadi-lo convenientemente, se a pilha for pouco usada, o *heap* pode invadi-la também. Essa é uma estratégia de otimização do tempo em que memória era um recurso caro, mas permaneceu por questões de compatibilidade com código legado.

A área estática no "fundo" da pilha (na verdade, em cima da pilha, mas como ela cresce ao contrário, é como se fosse o fundo no sentido lógico) ocupa uma posição conveniente, pois ela aparece como uma camada adicional da pilha.

Há uma representação esquemática da divisão da memória na resposta mostrada na página 5.

(b) A evolução dos dados ao longo da execução do programa para o escopameno dinâmico é mostrada na página 5.

O valor impresso é **35**. O ambiente de referenciamento na primeira execução de `h` é:

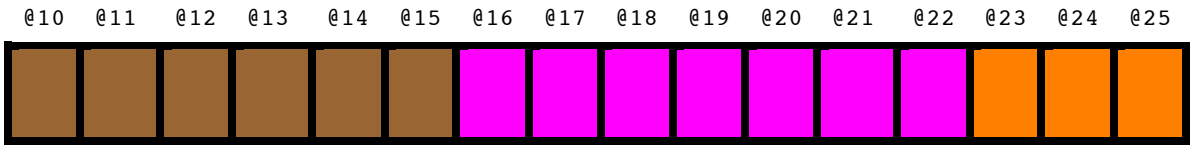
Variável	Definida em
a	g
b	main
d	g
e	g
i	local
k	local

(c) Os escopos das variáveis do programa são mostrados na página 7. A evolução dos dados, na página 8.

O valor impresso é **18**. O ambiente de referenciamento de `h` sempre é:

Variável	Definida em
a	g
b	global
d	g
e	g
i	local
k	local

(d) Sim. Quando o programa termina, há um bloco alocado no *heap* que não foi liberado e não há nenhum ponteiro para o bloco mediante o qual seja possível realizar sua liberação.



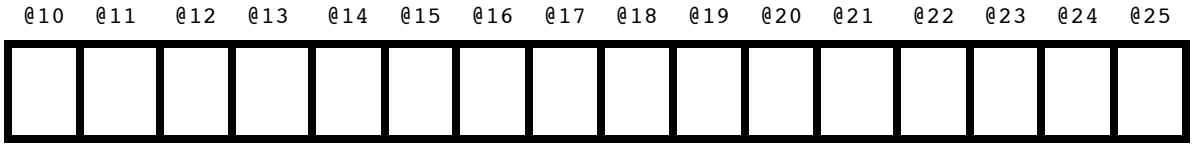
t0	20	-12	14	205	27	11	40	41	52	-2	-39	-198	4	**c** 100	**b** -10	**a** 34
t1	**m** 12	**v** @0	14	205	27	11	40	41	52	-2	-39	**main** **b** 3	**d** @10	**</** 100	* -10	18
t2	12	@0	14	205	27	11	40	41	52	-2	**f** **i** 21	19	@10	-3	* -10	18
t3	12	@0	14	205	27	11	40	41	52	-2	21	19	@10	**</** -3	* -10	15
t4	12	@0	**m** 14	**v** 205	27	11	40	41	**d** @12	**g** **a** 19	**e** @10	19	* @10	**</** -3	* -10	* 15
t5	12	@0	14	205	27	11	40	**f** **i** -6	@12	19	@10	20	* @10	-3	* -10	* 15
t6	12	@12	33	@0	27	11	**h** **i** @12	**k** @10	@12	19	@10	20	* @10	**</** -3	* -10	* 15
t7	12	@12	33	@0	27	11	@12	@10	@12	19	@10	20	@10	**</** -3	* -10	15

Resposta para o escopamento **dinâmico**.

- Nomes de funções e variáveis são comentários inseridos no desenho (em **azul**). Eles não são conteúdos de células de memória. O valor `@0` corresponde a `null`.
- Asteriscos (*) indicam uma posição de memória que não é visível na função em execução naquele instante devido ao sombreamento causado por outra variável na pilha.
- Asteriscos duplos (**) indicam uma posição de memória que não é visível na função em execução naquele instante por outros motivos.

A legenda de cores é a seguinte:

- Valor em cinza – sobra de uso anterior da célula de memória.
- Valor em vermelho – valor alterado na janela de tempo corrente.
- Célula em vermelho – alocação dinâmica em *heap*.
- Célula em cinza – desalocação em *heap*.

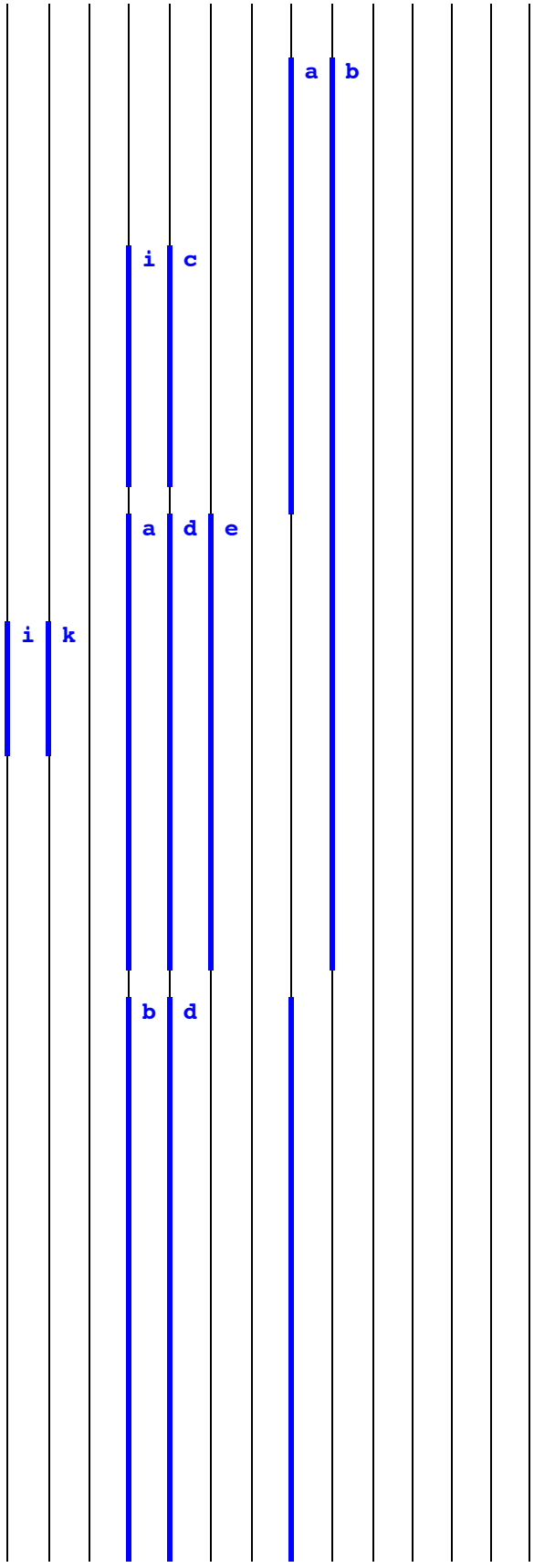


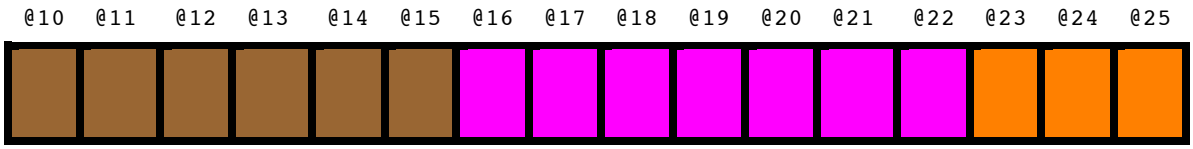
HEAP | **PILHA** | **Área Estática**

t0														**c** 100	**b** -10	**a** 34
t1	**m** 12	**v** @0										**main** **b** 3	**d** @10	** "	* "	18
t2	"	"									**f** **i** 21	19	"	-3	* "	"
t3	"	"										"	"	** "	* "	15
t4	"	"	**m**	**v**					**d** @12	**g** **a** 19	**e** @10	"	* "	** "	* "	* "
t5	"	"						**f** **i** -6	"	"	"	20	* "	"	* "	* "
t6	"	@12	33	@0			**h** **i** @12	**k** @10	"	"	"	"	* "	** "	* "	* "
t7			"	"								"	"	** "	* "	"

VISÃO DA SOLUÇÃO FEITA À CANETA NUMA FOLHA DE RESPOSTA.
O QUE ESTÁ EM AZUL REPRESENTA ALGO FEITO À CANETA.

```
1  #include <stdio.h>
2
3  int a = 34, b = -10;
4
5  typedef struct _w {
6        int m;
7        struct _w *v:
8  } W;
9
10 void f() {
11       int i;
12       static int c = 100;
13
14       i = a + b;
15       if (c > 0)
16             scanf("%d", &c);
17       b = a++;
18 }
19
20 void g(W *e) {
21       int a;
22       W *d;
23
24       void h(W *i, W *k) {
25             i->m = (k->m)++;
26             i->v = NULL;
27             k->v = i;
28       }
29
30       a = b;
31       d = (W*) malloc(sizeof(W));
32
33       f();
34
35       h(d, e);
36 }
37
38 void main(int argc, char* argv[]) {
39       int b;
40       W *d;
41
42       a = 18;
43       b = 3;
44       d = (W*) malloc(sizeof(W));
45       d->m = 33;
46       d->v = NULL;
47
48       if (read() > b)
49             f();
50
51       scanf("%d", &a);
52
53       g(d);
54
55       free(d);
56
57       printf("%d", a + b);
58 }
```





t0	20	-12	14	205	27	11	40	41	52	-2	-39	-198	4	**c** 100	**b** -10	**a** 34
t1	**m** 12	**v** @0	14	205	27	11	40	41	52	-2	-39	**main** **b** 3	**d** @10	100	-10	18
t2	12	@0	14	205	27	11	40	41	52	-2	**f** **i** 8	3	@10	-3	19	18
t3	12	@0	14	205	27	11	40	41	52	-2	8	3	@10	-3	19	15
t4	12	@0	**m** 14	**v** 205	27	11	40	41	**d** @12	**g** **a** 19	**e** @10	3	@10	-3	19	15
t5	12	@0	14	205	27	11	40	**f** **i** 34	@12	19	@10	3	@10	-3	16	15
t6	12	@12	33	@0	27	11	**h** **i** @12	**k** @10	@12	19	@10	3	@10	-3	16	15
t7	12	@12	33	@0	27	11	@12	@10	@12	19	@10	3	@10	-3	16	15

Resposta para o escopamento **estático**.

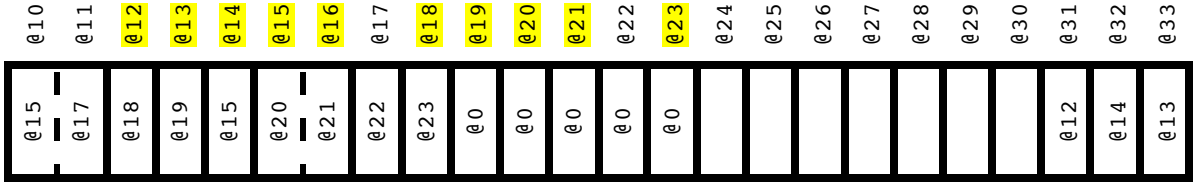
A evolução da pilha foi igual à do escopamento dinâmico **por coincidência.**

As variáveis visíveis em cada instante são definidas por um esquema mais complexo que coincide com aquele definido no próprio código fonte, portanto não faz sentido indicar variáveis visíveis no esquema.

Resposta da questão 3

Fase *Mark*

Começando pela base da pilha (@33), marcam-se:
@33 → @13 @19
@32 → @14 @15 @16 @20 @21
@31 → @12 @18 @23



Fase *Sweep*