

SSC0600 - Introdução à Ciência de Computação I  
 Tópico: Estruturas de Repetição e Tipos de Dados  
 Compostos (Strings e Vetores)

Provinha 2(a)

25 de abril de 2017

N.º USP:

<input type="text"/>	0	<input type="text"/>	0	<input type="text"/>	0	<input type="text"/>	0	<input type="text"/>	0	<input type="text"/>	0	<input type="text"/>	0
<input type="text"/>	1	<input type="text"/>	1	<input type="text"/>	1	<input type="text"/>	1	<input type="text"/>	1	<input type="text"/>	1	<input type="text"/>	1
<input type="text"/>	2	<input type="text"/>	2	<input type="text"/>	2	<input type="text"/>	2	<input type="text"/>	2	<input type="text"/>	2	<input type="text"/>	2
<input type="text"/>	3	<input type="text"/>	3	<input type="text"/>	3	<input type="text"/>	3	<input type="text"/>	3	<input type="text"/>	3	<input type="text"/>	3
<input type="text"/>	4	<input type="text"/>	4	<input type="text"/>	4	<input type="text"/>	4	<input type="text"/>	4	<input type="text"/>	4	<input type="text"/>	4
<input type="text"/>	5	<input type="text"/>	5	<input type="text"/>	5	<input type="text"/>	5	<input type="text"/>	5	<input type="text"/>	5	<input type="text"/>	5
<input type="text"/>	6	<input type="text"/>	6	<input type="text"/>	6	<input type="text"/>	6	<input type="text"/>	6	<input type="text"/>	6	<input type="text"/>	6
<input type="text"/>	7	<input type="text"/>	7	<input type="text"/>	7	<input type="text"/>	7	<input type="text"/>	7	<input type="text"/>	7	<input type="text"/>	7
<input type="text"/>	8	<input type="text"/>	8	<input type="text"/>	8	<input type="text"/>	8	<input type="text"/>	8	<input type="text"/>	8	<input type="text"/>	8
<input type="text"/>	9	<input type="text"/>	9	<input type="text"/>	9	<input type="text"/>	9	<input type="text"/>	9	<input type="text"/>	9	<input type="text"/>	9

← Por favor codifique seu Número USP na esquerda e escreva seu nome abaixo.

Nome e sobrenome:

.....

.....

**Question [remember-unistructural] ♣** Marque (X) nas opções que são usadas para representar as **estruturas de repetição** (*loops*) em pseudocódigo ou na Linguagem C.

☒ while (*expressão*) { ... }

☒ for (*expressão*; *expressão*; *expressão*) { ... }

☒ do { ... } while (*expressão*);

☐ if (*expressão*) { ... }

☐ if (*expressão*) { ... } else { ... }

☐ if-not (*expressão*) { ... }

☐ if-not (*expressão*) { ... } else { ... }

☐ while (*expressão*) { ... } other-case { ... }

☐ do { ... } until (*expressão*) { ... }

☒ PARA *expressão* ATÉ *expressão* PASSO *expressão* FAÇA ... FIMPARA

☒ ENQUANTO *expressão* FAÇA ... FIMENQUANTO

☒ REPITA *expressão* ATÉ *expressão*

☐ SE *expressão* ENTÃO ...

☐ SE *expressão* ENTÃO ... SENÃO ...

☐ SENÃO *expressão* ENTÃO ...

☐ SENÃO *expressão* ENTÃO ... SENÃO ...

☐ PARA *expressão* ATÉ *expressão* FAÇA ... FIMPARA

☐ ENQUANTO *expressão* FAÇA ... EMOUTROSCASOSFAÇA ... FIMENQUANTO

☐ Nenhuma das alternativas está correta

**Question [understand-relational] ♣** Marque (X) nas afirmativas verdadeiras em relação aos trechos de código em Linguagem C apresentados nas Listagem 1, 2, 3 e 4

- *Observação:* `arr[n][n]` é a matriz  $n \times n$  de números inteiros

- ☒ Listagem 1 gira 180 graus no sentido horário ou anti-horário a matriz `arr[n][n]`
- ☒ Listagem 2 gira 90 graus no sentido horário a matriz `arr[n][n]`
- ☒ Listagem 3 gira 360 graus no sentido horário ou anti-horário a matriz `arr[n][n]`
- ☒ Listagem 4 gira 90 graus no sentido anti-horário a matriz `arr[n][n]`
- ☐ Listagem 1 gira 360 graus no sentido horário ou anti-horário a matriz `arr[n][n]`
- ☐ Listagem 2 gira 270 graus no sentido horário a matriz `arr[n][n]`
- ☐ Listagem 4 gira 270 graus no sentido anti-horário a matriz `arr[n][n]`
- ☐ Listagem 2 gira 90 graus no sentido anti-horário a matriz `arr[n][n]`
- ☐ Listagem 4 gira 90 graus no sentido horário a matriz `arr[n][n]`
- ☐ Listagem 1 gira 90 graus no sentido horário a matriz `arr[n][n]`
- ☐ Listagem 2 gira 360 graus no sentido horário ou anti-horário a matriz `arr[n][n]`
- ☐ Listagem 3 gira 90 graus no sentido anti-horário a matriz `arr[n][n]`
- ☐ Listagem 4 gira 360 graus no sentido horário ou anti-horário a matriz `arr[n][n]`
- ☐ *Nenhuma das alternativas está correta*

**Question [apply-multistructural-1] ♣** Marque (X) na saída do programa apresentado na Listagem 6.

- ☒
  - 0 1 2
  - 0 1 2
  - 0 1 2
  - 0 0 0
  - 1 1 1
  - 2 2 2
  - 2 1 0
  - 2 1 0
  - 2 1 0
- ☐
  - 2 2 2
  - 2 2 2
  - 2 2 2
- ☐ *Nenhuma das alternativas está correta*

**Question [apply-multistructural-2] ♣** Marque (X) nas afirmativas verdadeiras em relação ao programa da Listagem 7.

- ☒ Depois que o código for executado,  $x$  contém os valores:  $\{5, 4, 3, 4, 2, 0\}$
- ☐ Depois que o código for executado,  $x$  contém os valores:  $\{5, 4, 2, 3, 2, 0\}$
- ☐ Depois que o código for executado,  $x$  contém os valores:  $\{10, 8, 6, 2, 1, 0\}$
- ☐ Depois que o código for executado,  $x$  contém os valores:  $\{10, 8, 2, 3, 1, 0\}$
- ☒ Depois que o código for executado,  $y$  contém os valores:  $\{5, 2, 6, 4, 4, 0\}$
- ☐ Depois que o código for executado,  $y$  contém os valores:  $\{5, 2, 4, 3, 4, 0\}$
- ☐ Depois que o código for executado,  $y$  contém os valores:  $\{5, 1, 6, 4, 8, 0\}$
- ☐ Depois que o código for executado,  $y$  contém os valores:  $\{5, 1, 2, 6, 8, 0\}$
- ☐ *Nenhuma das alternativas está correta*

**Question [evaluate-multistructural] ♣** Marque (X) nas afirmativas verdadeiras em relação ao programa da Listagem 7.

- ☒ A condição `i1 < j1` da Linha 9 é avaliada 4 vezes
- ☐ A condição `i1 < j1` da Linha 9 é avaliada 3 vezes
- ☐ A condição `i1 < j1` da Linha 9 é avaliada 6 vezes
- ☒ A condição `j2 > j2` da Linha 16 é avaliada 4 vezes
- ☐ A condição `j2 > j2` da Linha 16 é avaliada 3 vezes
- ☐ A condição `j2 > j2` da Linha 16 é avaliada 6 vezes
- ☒ O código na estrutura de repetição externa (*outer loop*), linhas 10 até 25, é repetido 3 vezes
- ☐ O código na estrutura de repetição externa (*outer loop*), linhas 10 até 25, é repetido 4 vezes
- ☐ O código na estrutura de repetição externa (*outer loop*), linhas 10 até 25, é repetido 6 vezes
- ☒ O código na estrutura de repetição interna (*inner loop*), linhas 17 até 21, é repetido 3 vezes
- ☐ O código na estrutura de repetição interna (*inner loop*), linhas 17 até 21, é repetido 4 vezes
- ☐ O código na estrutura de repetição interna (*inner loop*), linhas 17 até 21, é repetido 6 vezes
- ☐ Nenhuma das alternativas está correta

**Question [analyse-relational-1] ♣** Marque (X) nas modificações que, de maneira independente umas das outras, façam com que o programa apresentado na Listagem 6 imprima:

```
0 0 0
1 1 1
2 2 2
```

- ☒ A linha 14 deve ser mudada para: `printf("%d ", m[j][i]);`
- ☒ A linha 9 deve ser mudada para: `m[row][col] = row;`
- ☒ A linha 9 deve ser mudada para: `m[col][row] = col;`
- ☐ A linha 12 deve ser mudada para: `for (i=2; i>=0; i--) {`
- ☐ A linha 13 deve ser mudada para: `for (j=2; j>=0; j--) {`
- ☒ A linha 13 deve ser mudada para: `for (j=2; j>=0; j--) {`  
A linha 14 deve ser mudada para: `printf("%d ", m[j][i]);`
- ☐ A linha 12 deve ser mudada para: `for (i=2; i>=0; i--) {`  
A linha 14 deve ser mudada para: `printf("%d ", m[j][i]);`
- ☐ Nenhuma das alternativas está correta

**Question [analyse-relational-2] ♣** O trecho de código apresentado na Listagem 5 tem sido proposto para efetuar a ordenação ascendente (de menor a maior) de um vetor de inteiros `arr[n]` (array `arr` de tamanho `n`). No entanto, o programa não funciona adequadamente. Marque (X) nas modificações necessárias que, em conjunto, façam o programa funcionar adequadamente.

- ☒ A linha 5 deve ser mudada para: `if (arr[j] < arr[m]) {`
- ☐ A linha 5 deve ser mudada para: `if (arr[j] > arr[m]) {`
- ☐ A linha 5 deve ser mudada para: `if (arr[j] >= arr[m]) {`
- ☒ A linha 4 deve ser mudada para: `while (j < n) {`
- ☐ A linha 4 deve ser mudada para: `while (j < n-1) {`
- ☐ A linha 4 deve ser mudada para: `while (j > n) {`
- ☐ A linha 4 deve ser mudada para: `while (j >= n-1) {`
- ☐ Nenhuma das alternativas está correta