**ATIVIDADE 02**

**1.1. Analise a estrutura de dados esquematizada a seguir:**



**Explique por que essa estrutura:**

**a) Não é um conjunto.**

**R:** Porque Conjunto é uma coleção em que não há ordem nem repetição. Na figura observa-se que existe uma ordem de relacionamento.

**b) Não é uma lista.**

**R:** Lista é uma coleção de itens linearmente. Cada item em uma lista em um único predecessor e um único sucessor. Exceto o primeiro item, que não tem predecessor e o último item, que não tem sucessor.

Na figura, observa-se que:

* Item ‘c’ possui 2 predecessores e nenhum sucessor
* O item ‘d’ não possui predecessor e 2 sucessores

**c) Não é uma árvore.**

**R:** Uma árvore é uma coleção de itens organizados hierarquicamente. Cada item em uma árvore tem um único predecessor e vários sucessores, exceto a raiz, que não tem predecessor, e as folhas, que não têm sucessores.

Na figura, observa-se que os itens:

* Possuem apenas um sucessor: ‘a’, ‘b’, ‘e’
* Possuem mais de um predecessor: ‘c
* Não possuem predecessor: ‘a’, ‘d’

**1.2. Indique a estrutura de dados mais adequada para ser usada num programa que precisa representar:**

**a) Estradas entre cidades de um mapa.**

**R:** Grafo. Pois cada cidade se relacionar com várias outras através das estradas, se assemelhando a uma organização em rede.

**b) A estrutura organizacional de uma empresa.**

**R:** Árvore. Pois a organização de uma empresa é hierárquica, ou seja, a partir do dono (ou presidente) os cargos/funções vão se ramificando hierarquicamente. Cada indivíduo é subordinado a um chefe.

**c) A fila de impressão em um sistema operacional.**

**R:** Lista. Pois na fila de impressão existe uma sequencia a ser seguida, onde as ordens de impressão são executadas uma por uma.

**d) A estrutura de diretórios em um sistema operacional.**

**R:** Árvore. Pois o sistema de diretórios de um sistema operacional é organizado de forma hierárquica. A partir da raiz, são criadas novas pastas. Cada pasta pode conter várias outras, mas só pode ter uma predecessora.

**1.3. Ao ser executado, o programa a seguir exibe a palavra *diferentes*. Explique porque isso acontece.**

#include <stdio.h>

Int main (void) {

char s [3] = “um”;

char t [3] = “dois”;

if ( s == t ) puts ( “iguais” );

else puts ( “diferentes”);

return 0;

}

**R:** As variáveis que representam os vetores são ponteiros para os seus primeiros elementos. Por isso, quando compara s == t , os ponteiros apontam para o mesmo local de memória de cada um, ou seja, diferentes.

**1.4. Explique o que acontece quando o programa a seguir é executado.**

#include <stdio.h>

typedef struct { char valor[10]; } Str;

int main () {

Str x = { "um" };

Str y = { "dois" };

puts ( x.valor ); *// é impresso “um”*

x = y; *// x passa a ter ”dois”. Pois abas as variáveis têm apenas uma posição/elemento*

puts ( x.valor );  *// é impresso “dois”*

return 0;

}

**R:** É retornado os caracteres “um” e “dois”. Ocorre que

**1.5. Explique o que acontece quando o programa a seguir é executado.**

*// Sabe-se que um ponteiro guarda a endereço de memória de uma variável*

*// Para acessar a variável apontada por um ponteiro, basta prefixar com \**

*// Então, p é um ponteiro, \*p é a variável que ele aponta*

#include <stdio.h>

int main (void) {

int a = 3, b = 5;

int \*p = &a, \*q = &b; *// p aponta para a, q aponta para b*

\*p = \*p + \*q; *// a = 8, b = 5*

\*q = \*p - \*q;  *// a = 8, b = 3*

\*p = \*p - \*q; *// a = 5, b = 3*

printf ( “%d, %d \n”, a, b ); *// imprime na tela o conteúdo das variáveis a, b*

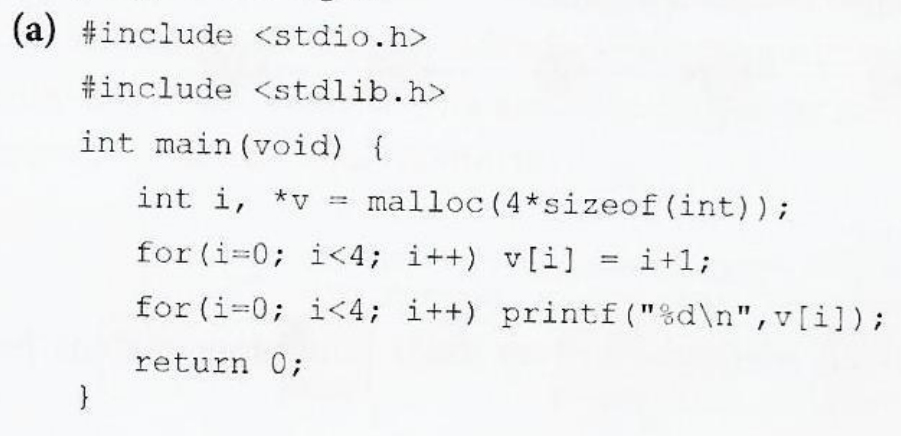
return 0;

}

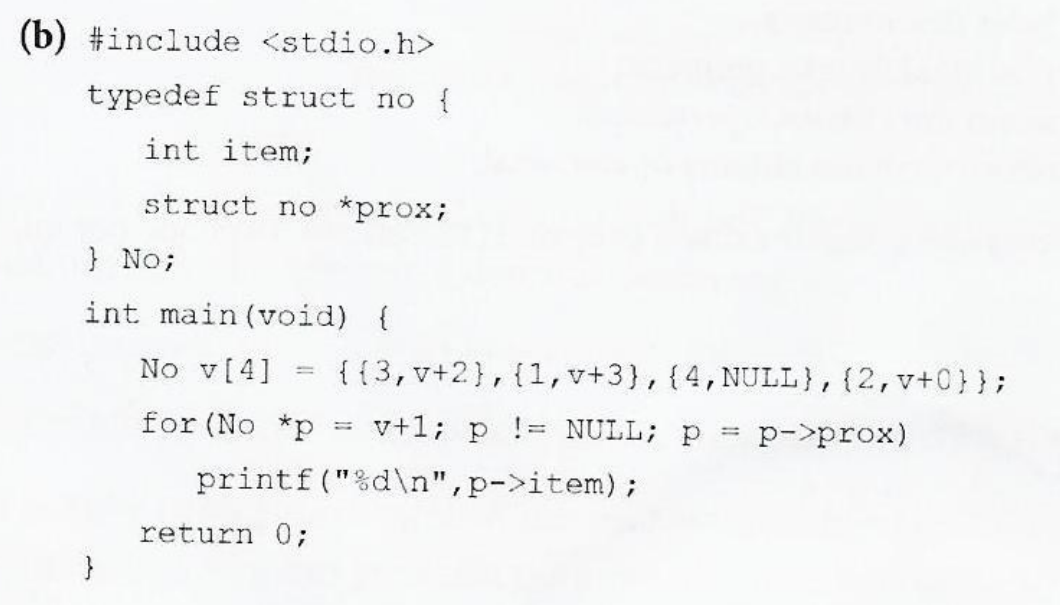
**R:** O programa retorna: **5, 3**.

**1.6. Indique a forma de alocação de memória (*estática sequencial, estática encadeada, dinâmica sequencial ou dinâmica encadeada*) usada para armazenar os itens 1, 2, 3 e 4, em cada um dos programas a seguir:**

**a)** Alocação dinâmica sequencial, pois o uso do *malloc* representa a alocação dinâmica, e os dados são inseridos em sequência com o uso do *for*



**b)** Alocação estática encadeada, com o uso de vetores e do *struct* próprio do encadeamento



**c)** Dinâmica encadeada, com o uso do *malloc* e do *struct* próprio do encadeamento

