UFMT - Universidade Federal de Mato Grosso

Disciplina: Algoritmos e Programação II
Avaliação de Aprendizagem

Nome do Estudante: _____

PROVA P1

Leia atentamente as questões antes de responder.

---

1. (Valor: 1,0) [UFMT-2017] Na linguagem C, é possível realizar alocações de memória utilizando alocação dinâmica ou estática. Assinale a alternativa que representa uma alocação dinâmica de um vetor do tipo primitivo `double` com 10 posições na linguagem C.

   A. `malloc(10, sizeof(double))`

   B. `double[10]`

   C. `double * 10`

   D. `malloc(10 * sizeof(double))`

   E. Nenhuma das alternativas anteriores está correta.

2. (Valor: 1,0) [VUNESP-2022 (adaptada)] Na linguagem C, estruturas (definidas pela palavra-chave `struct`) podem ser passadas como parâmetros em chamadas de funções. Nesse contexto, assinale a alternativa correta.

   A. Por padrão, estruturas são passadas por referência.

   B. Estruturas só podem ser passadas como parâmetro por meio do uso de ponteiros.

   C. Passar estruturas por referência é mais eficiente que passá-las por valor, uma vez que a passagem por valor requer que a estrutura inteira seja copiada.

   D. É necessário o uso da palavra-chave `inline` na declaração da função que recebe uma estrutura como parâmetro.

   E. Apenas funções do tipo `void` podem receber estruturas como parâmetros.

3. (Valor: 1,0) [FCC-2010] Na linguagem C, considere:

   I. O endereço armazenado em um ponteiro deve ser do mesmo tipo que o ponteiro (ex. um ponteiro para um `int` não pode armazenar o endereço de um `float`).

   II. Exceção à regra apontada em (I) é o ponteiro `void`.

   III. Não é possível chamar uma função segundo seu endereço, ainda que por meio de um ponteiro que armazena o endereço de início dessa função.

   Está correto o que se afirma em:

   A. I, apenas.

   B. II, apenas.

   C. I e II, apenas.

D. II e III, apenas.

E. I, II e III.

4. (Valor: 1,0) Roberval está desenvolvendo um trabalho de programação em dupla com o seu amigo João. Ele declarou um vetor da seguinte forma:

```
int vetor[4];

```

João, entretanto, pegou o código e alterou essa linha, deixando-a da seguinte forma:

```
int *vetorAlocDinamica = (int*) calloc(4, sizeof(int));

```

Considerando as declarações de Roberval e João para a criação de um vetor em C, qual das seguintes afirmações é verdadeira?

A. Ambas as declarações criam um vetor de inteiros de tamanho fixo na memória, e o espaço alocado não pode ser modificado após a inicialização.

B. A declaração de João usa `calloc`, o que significa que o vetor é inicializado com zeros, enquanto a declaração de Roberval inicializa automaticamente o vetor com zeros.

C. Ambas as declarações resultam em vetores cujos tamanhos não podem ser alterados em tempo de execução usando a função realloc.

D. A declaração de João é inválida em C, pois a alocação dinâmica de memória não é suportada pela linguagem.

E. A declaração de Roberval aloca memória na stack, enquanto a declaração de João aloca memória no heap.

5. (Valor: 1,0) Bartolomeu é um estudante de programação que está aprendendo a lidar com aritmética de ponteiros. Atualmente, o seu código está no seguinte estágio de implementação:

```
#include <stdio.h>

int main(void){
    int vetor[4] = {1, 2, 3, 4};
    printf("%d", ________ );
    return 0;
}

```

Agora, confira as afirmações a seguir:

I. Se o trecho que completa a lacuna for `*vetor+0`, a saída do programa será 1.

II. Se o trecho que completa a lacuna for `*vetor+3`, a saída do programa será 4.

III. Se o trecho que completa a lacuna for `vetor+0`, a saída do programa será 1.

Assinale a alternativa que apresenta corretamente a ordem de classificação das afirmações anteriores como verdadeiras (V) ou falsas (F).

A. F - F - F

B. V - V - V

C. V - V - F

D. V - F - F

E. Nenhuma das opções anteriores.

6. (Valor: 1,0) Bartolomeu e seu amigo Franciscleidson estão aprendendo, juntos, a manipular arquivos em C. Bartolomeu implementou um algoritmo que escreveu um arquivo codificado e o enviou para Franciscleidson. O desafio deste era implementar um algoritmo que não apenas lesse o arquivo, mas também fosse capaz de decodificar a mensagem. No entanto, para a surpresa de Franciscleidson, o conteúdo do arquivo sumiu quando ele executou o seu programa. O que pode ter acontecido nessa situação?

   A. Franciscleidson usou o modo de leitura "r"ao abrir o arquivo, o que automaticamente apaga seu conteúdo antes da leitura.

   B. Franciscleidson acidentalmente abriu o arquivo no modo de escrita “w”, o que apaga o conteúdo do arquivo ao abri-lo.

   C. Franciscleidson usou o modo “rb” para ler o arquivo, o que é incorreto porque este modo é usado apenas para arquivos binários, não afetando o conteúdo.

   D. O programa de Franciscleidson falhou ao tentar abrir o arquivo no modo “r+”, mas isso não deveria apagar o conteúdo do arquivo, apenas permite leitura e escrita.

   E. Franciscleidson utilizou corretamente o modo “a” para adicionar conteúdo ao final do arquivo, o que não deveria apagar seu conteúdo existente.

7. (Valor: 1,0) Pedro está desenvolvendo um sistema em C para gerenciar informações sobre livros em uma biblioteca. Cada livro possui um título, autor e ano de publicação. Pedro decide usar uma struct para representar cada livro e um vetor de structs alocado dinamicamente para armazenar os dados de múltiplos livros. Considere a seguinte declaração da struct:

```c
typedef struct {
    char titulo[100];
    char autor[50];
    int ano;
} Livro;

```

   Pedro deseja criar um vetor dinâmico de 5 livros. Qual das seguintes linhas de código realiza corretamente essa alocação dinâmica?

   A. `Livro *vetorLivros = (Livro*) calloc(5, sizeof(Livro));`

   B. `Livro vetorLivros[5];`

   C. `Livro *vetorLivros = (Livro*) malloc(5);`

   D. `Livro *vetorLivros = calloc(5, sizeof(Livro*));`

   E. Nenhuma das opções anteriores está correta.

8. (Valor: 1,0) Considere as definições das structs A, B e C apresentadas abaixo.

```c
typedef struct {
    int a;
    int b;
    char c;
    char d;
} A;

```

```c
typedef struct {
    int a;
    char b;
    char c;
    int d;
} B;

typedef struct {
    char a;
    int b;
    char c;
    int d;
} C;

```

Supondo um compilador típico onde `int` ocupa 4 bytes e `char` ocupa 1 byte, qual das seguintes afirmações sobre o espaço ocupado por cada estrutura é verdadeira?

A. A estrutura C ocupa mais espaço na memória do que a estrutura A.

B. As estruturas A, B e C ocupam, respectivamente, 10, 10 e 10 bytes de espaço na memória.

C. A estrutura A ocupa menos espaço na memória do que a estrutura B.

D. Todas as estruturas ocupam exatamente o mesmo espaço na memória.

E. As estruturas A, B e C ocupam, respectivamente, 12, 12 e 12 bytes de espaço na memória devido ao alinhamento de memória.

9. (Valor: 1,0) Demerval está desenvolvendo um programa em C que manipula um vetor de structs. Ele deseja implementar uma função/procedimento capaz de simular a exclusão de um registro, como visto nas aulas de Algoritmos e Estruturas de Dados. Considere a seguinte declaração:

```c
#include <stdio.h>
#include <stdlib.h>
#define TAM 5

typedef struct {
    char nome[30];
    char curso[10];
} Disciplina;

int main(void) {
    Disciplina d[TAM];
    int pos = 0;
}

```

Nesse cenário, considerando que nenhuma leitura de dados do teclado deve ser feita pela função a ser implementada, qual seria o correto protótipo/assinatura da função/procedimento?

A. `void excluir(Disciplina disc[], int *pos)`

B. `void excluir(Disciplina disc, int *pos, int posExc)`

C. `void excluir(Disciplina disc[], int *pos, int posExc)`

D. `void excluir(Disciplina disc[5], int pos, int posExc)`

E. Nenhuma das opções anteriores apresenta a opção correta.

10. (Valor: 1,0) Em linguagem de programação C, qual é a principal diferença entre uma struct e uma union?

    A. struct e union são idênticas; ambas permitem que diferentes tipos de dados sejam armazenados no mesmo endereço de memória ao mesmo tempo.

    B. Em uma struct, todos os membros compartilham o mesmo espaço de memória, enquanto em uma union, cada membro ocupa seu próprio espaço de memória, permitindo que todos os membros sejam acessados ao mesmo tempo.

    C. Uma struct aloca espaço de memória suficiente para armazenar todos os seus membros juntos, enquanto uma union aloca espaço de memória suficiente para o maior de seus membros, com todos os membros compartilhando este espaço.

    D. struct é usada exclusivamente para tipos de dados primitivos, enquanto union é usada para tipos de dados compostos e complexos.

    E. Em uma union, o compilador automaticamente inicializa todos os membros, enquanto em uma struct, a inicialização de cada membro deve ser explicitamente definida pelo programador.