**Exercícios 8-20 – Alocação de memória**

**8)** No malloc, considera-se apenas o número de bytes para fazer a alocação de memória. Já o calloc, utiliza o número de elementos e o tamanho deles para a alocação.

Além disso, o calloc zera todos os bits de memória alocada enquanto o malloc não.

**9)** Caso não haja necessidade de zerar a memória alocada, o malloc é recomendável por ter mais rapidez.

**10)** O tamanho de memória alocada será definida com o uso de malloc ou calloc, considerando o número de bytes.

Terá sucesso quando o ponteiro retornar para o início do intervalo e erro quando retornar NULL, o que significa que não há espaço de memória disponível.

**11)** O realloc tem como principal objetivo, alterar o tamanho de um bloco que já foi alocado, preservando o conteúdo dentro dele.

**12)** Alocação de memória estática ocorre na declaração de uma variável, de forma automática pelo compilador.

Alocação de memória dinâmica ocorre durante a execução do programa, de forma manual, permitindo ao usuário usar espaços de memória de tamanho arbritário, usar estrutura de dados com encadeamento.

1**3)** Na alocação estática, as variáveis alocadas possuem um tamanho fixo e estão alocadas sequencialmente na memória, só serão excluídas com o fim da execução do programa.

Já na alocação dinâmica, as variáveis alocadas não precisam ter um tamanho fixo, não precisam estar organizadas de forma sequencial e podem ser liberadas durante alguma parte no programa, quando não são mais necessárias e não necessariamente com o fim da execução como ocorre na alocação de memória estática.

**14)** Alocação automática de memória ocorre durante a alocação estática, quando as variáveis são declaradas e o programa automaticamente salva uma posição na memória.

**15)**Quando não se consegue alocar memória, o valor retornado por malloc é NULL, por isso sempre é necessário verificar se a alocação ocorreu com sucesso, evitando possíveis erros futuros.

**16)** Vazamento de memória ocorre quando um programa de computador não gerencia corretamente suas alocações de memória, em que a memória não é liberada quando não é mais necessária, ou quando um objeto armazenado não pode mais ser acessado pelo código em execução.

Ela está relacionada a erros de programação e pode levar a falhas no sistema se a memória for completamente consumida.

**17)** Fragmentação é o desperdício de espaço disponível em memória.

A fragmentação interna se baseia na perda de espaço dentro de uma área de tamanho fixo, quando um arquivo não ocupa toda a área destinada a ele.

A fragmentação externa, baseada no particionamento dinâmico, ocorre quando os programas vão terminando e deixando um espaço cada vez menor na memória, não permitindo o ingresso de novos programas.

**18)** A função free tem como principal objetivo liberar a memória alocada após o uso.

Ela é utilizada junto com a indicação do ponteiro para o início do intervalo de memória a ser liberado.

Seu uso é bem importante para evitar problemas como o vazamento de memória.

**19)**Um ponteiro pendente é aquele que não possui um endereço associado.

Seu valor não é um endereço válido e sua utilização pode danificar partes diferentes do sistema.

**20)**Alguns erros comuns no gerenciamento de memória:

-Esquecer de alocar memória e tentar acessar o conteúdo da variável;

-Esquecer de desalocar memória;

-Tentar acessar o conteúdo da variável após desalocá-la.