ATIVIDADE PARA AVALIAÇÃO - SEMANA 3

A resposta correta da questão está identificada com a cor Vermelha.

1. **(1 ponto)**Sobre a linguagem C++, é correto afirmar que:
   1. É uma linguagem interpretada, então não existe a geração de um executável nativo para uma determinada plataforma.
   2. **Possui tipagem estática, então as variáveis não podem mudar de tipo após serem declaradas.**
   3. Programas escritos em C++ são mais lentos que programas escritos em Python, por esta última ser uma linguagem compilada.
   4. Não fornece suporte ao paradigma orientado a objetos.
   5. O escopo é delimitado pela indentação do código.

A resposta correta é: “Possui tipagem estática, então as variáveis não podem mudar de tipo após serem declaradas.”  
  
JUSTIFICATIVA  
  
É uma linguagem interpretada, então não existe a geração de um executável nativo para uma determinada plataforma.  
Incorreto, C++ é uma linguagem compilada.  
Correto.  
Programas escritos em C são mais lentos que programas escritos em Python, por esta última ser uma linguagem compilada.  
Incorreto, programas escritos em C++ são mais rápidos porque python é uma linguagem interpretada.  
Não fornece suporte ao paradigma orientado a objetos.  
Incorreto, o suporte à Programação Orientada a Objetos é fornecido.  
O escopo é delimitado pela indentação do código.  
Incorreto, o escopo é delimitado por chaves.

1. **(1 ponto)**Sobre os recursos e as regras de sintaxe da linguagem C++, assinale a opção correta.
   1. A sintaxe da linguagem C++ é semelhante à sintaxe da linguagem Python, ambas usam o operador ":" para delimitar o início do escopo.
   2. Tanto C++ quanto Python permitem manipulação explícita de ponteiros, o que torna ambas as linguagens igualmente capazes de acessar a memória principal do computador.
   3. Na passagem de parâmetro por valor, uma região de memória é passada para a função. Assim, qualquer mudança no valor dentro da função altera o valor da variável fora da função.
   4. A escolha da passagem de parâmetro por valor ou por referência importa por questões de desempenho, mas não altera o comportamento dos programas. Em outras palavras, se alguém trocar todas as passagens de parâmetro por valor para passagem de parâmetros por referência (e vice-versa), o programa executará da mesma maneira.
   5. **Para efetuar uma passagem de parâmetros por referência, colocamos o operador & antes do nome da variável.**

A resposta correta é: “Para efetuar uma passagem de parâmetros por referência, colocamos o operador & antes do nome da variável.”  
  
JUSTIFICATIVA  
  
A sintaxe linguagem C++ é semelhante à sintaxe da linguagem Python, ambas usam o operador ":" para delimitar o início do escopo.  
Incorreto, o operador ":" não inicia escopo em C++.  
Tanto C++ quanto Python permitem manipulação explícita de ponteiros, o que torna ambas as linguagens igualmente capazes de acessar a memória principal do computador.  
Incorreto, não há manipulação explícita de ponteiros em Python.  
Na passagem de parâmetro por valor, uma região de memória é passada para a função. Assim, qualquer mudança no valor dentro da função altera o valor da variável fora da função.  
Incorreto, o que foi mencionado na alternativa é a passagem de parâmetro por referência.  
A escolha de passagem de parâmetro por valor ou por referência importa por questões de desempenho, mas não altera o comportamento dos programas. Em outras palavras, se alguém trocar todas as passagens de parâmetro por valor para passagem de parâmetros por referência (e vice-versa), o programa executará da mesma maneira. Incorreto, trocar as passagens de parâmetro por valor para passagens por referência (e vice-versa) causará mudanças no comportamento dos programas.  
Para efetuar uma passagem de parâmetros por referência, colocamos o operador & antes do nome da variável. Correto.

1. **(1 ponto)** Sobre o trecho de código mostrado a seguir:

Assinale a alternativa verdadeira:

* 1. Ocorrerá um erro em tempo de compilação, porque intPointer não pode receber o valor retornado por &alpha.
  2. Não ocorrerá erro em tempo de compilação, mas o programa gerará erro em tempo de execução, porque intPoint não pode receber o valor retornado por &alpha.
  3. **A variável intPointer aponta para alpha. Nesse caso, alterações podem ser feitas diretamente usando alpha ou indiretamente com intPointer.**
  4. A variável alpha terá um valor diferente daquele conteúdo que está sendo apontado por intPointer na memória.
  5. A variável alpha é um ponteiro para uma região de memória, podendo acessar essa memória indiretamente.

A resposta correta é: “A variável intPointer aponta para alpha. Nesse caso, alterações podem ser feitas diretamente usando alpha ou indiretamente com intPointer.”  
  
JUSTIFICATIVA  
  
Ocorrerá um erro em tempo de compilação, porque intPointer não pode receber o valor retornado por &alpha.  
Incorreto, não será gerado erro em tempo de compilação neste código.  
Não ocorrerá erro em tempo de compilação, mas o programa gerará erro em tempo de execução, porque intPoint não pode receber o valor retornado por &alpha.  
Incorreto, não será gerado erro em tempo de execução.  
A variável alpha terá um valor diferente daquele que está sendo apontado por intPointer na memória.  
Incorreto, o valor de alpha será igual ao valor de \*intPointer.  
A variável intPointer aponta para alpha. Nesse caso, alterações podem ser feitas diretamente usando alpha ou indiretamente com intPointer.  
Correto.  
A variável alpha é um ponteiro para uma região de memória, podendo acessar essa memória indiretamente.  
Incorreto, alpha é uma variável convencional, não um ponteiro.

1. **(1 ponto)**Sobre alocação dinâmica, é correto afirmar que:
   1. **Alocação dinâmica permite alocar memória sem a necessidade de definir em tempo de compilação o tamanho a ser utilizado, dado que a alocação ocorre em tempo de execução.**
   2. O operador "new" para fazer alocação dinâmica em C++ permite gerar alocação de objetos, mas não permite fazer alocação de tipos pré-definidos, como "int" e "float".
   3. O operador "delete" permite desalocar qualquer espaço de memória, inclusive as regiões de memória alocadas estaticamente em tempo de compilação.
   4. Em C++, não há a necessidade de desalocar a memória com "delete", dado que a máquina virtual efetua essa tarefa periodicamente. A prática de desalocar memória é aconselhável, mas não essencial.
   5. Ao alocar espaço em uma região de memória com "new", temos a opção de acessar essa região com um ponteiro ou com uma variável convencional. Neste último caso, basta atribuir o retorno obtido com o comando "new" para uma variável que não é ponteiro.

A resposta correta é: “Alocação dinâmica permite alocar memória sem a necessidade de definir em tempo de compilação o tamanho a ser utilizado, dado que a alocação ocorre em tempo de execução.”  
  
JUSTIFICATIVA  
  
Alocação dinâmica permite alocar memória sem a necessidade de definir em tempo de compilação o tamanho a ser utilizado, dado que a alocação ocorre em tempo de execução.  
Correto.  
O operador "new" para fazer alocação dinâmica em C++ permite gerar alocação de objetos, mas não permite fazer alocação de tipos pré-definidos, como "int" e "float".  
Incorreto, a alocação de tipos pré-definidos é permitida.  
O operador "delete" permite desalocar qualquer espaço de memória, inclusive as regiões de memória alocadas estaticamente em tempo de compilação.  
Incorreto, é possível somente desalocar espaço de memória que foi alocado dinamicamente.  
Em C++, não há a necessidade de desalocar a memória com "delete", dado que a máquina virtual efetua essa tarefa periodicamente. A prática de desalocar memória é aconselhável, mas não essencial.  
Incorreto, não temos máquina virtual em C++ que faça algum tipo de coletor de lixo.  
Ao alocar espaço em uma região de memória com "new", temos a opção de acessar essa região com um ponteiro ou com uma variável convencional. Neste último caso, basta atribuir o retorno obtido com o comando "new" para uma variável que não é ponteiro.  
Incorreto, não é possível atribuir o retorno de "new" para uma variável que não é ponteiro.

1. **(1 ponto)**Sobre alocação de vetores em C++, é correto afirmar que:
   1. Em C++, um vetor alocado estaticamente ocupa regiões contíguas de memória, enquanto que um vetor alocado dinamicamente fica espalhado na memória de maneira não contígua.
   2. **Todos os elementos alocados em um vetor devem ser do mesmo tipo, e o tamanho do vetor deve ser informado antes de fazer a utilização do mesmo, não podendo ser aumentado ou diminuído posteriormente.**
   3. Em C++, é possível inicializar elementos no momento da declaração de um vetor estático usando uma lista inicializadora (lista de elementos entre chaves). A quantidade de elementos na lista não pode ser menor do que o tamanho declarado para o vetor, pois isso geraria erro em tempo de compilação.
   4. Em C++, é possível acessar um elemento usando a sintaxe de colchetes se quisermos acessar um único elemento, ou a sintaxe de chaves para acessar vários elementos ao mesmo tempo.
   5. Em C++, se o código tentar acessar uma posição que está fora dos limites do vetor, receberemos um erro em tempo de compilação dado que a informação do tamanho do vetor é conhecida antes de executar o programa.

A resposta correta é: “Todos os elementos alocados em um vetor devem ser do mesmo tipo, e o tamanho do vetor deve ser informado antes de fazer a utilização do mesmo, não podendo ser aumentado ou diminuído posteriormente."  
  
JUSTIFICATIVA  
  
Coloque aqui um comentário fundamentado ou a resolução passo a passo.  
Todos os elementos alocados em um vetor devem ser do mesmo tipo, e o tamanho do vetor deve ser informado antes de fazer a utilização do mesmo, não podendo ser aumentado ou diminuído posteriormente.  
Correto.  
Em C++, um vetor alocado estaticamente ocupa regiões contíguas de memória, enquanto que um vetor alocado dinamicamente fica espalhado na memória de maneira não contígua.  
Incorreto, um vetor alocado dinamicamente também ocupa posições contíguas de memória.  
Em C++, é possível inicializar elementos no momento da declaração de um vetor estático usando uma lista inicializadora (lista de elementos entre chaves). A quantidade de elementos na lista não pode ser menor do que o tamanho declarado para o vetor, pois isso geraria erro em tempo de compilação.  
Incorreto, é possível colocar menos elementos na lista.  
Em C++, é possível acessar um elemento usando a sintaxe de colchetes se quisermos acessar um único elemento, ou a sintaxe de chaves para acessar vários elementos ao mesmo tempo.  
Incorreto, não existe a referida sintaxe de chaves para acessar vários elementos ao mesmo tempo.  
Em C++, se o código tentar acessar uma posição que está fora dos limites do vetor, receberemos um erro em tempo de compilação dado que a informação do tamanho do vetor é conhecida antes de executar o programa.  
Incorreto, é função do programador gerenciar os limites do vetor, não havendo erro em tempo de compilação se o limite for violado.

1. **(1 ponto)** Sobre a estrutura de dados pilha com implementação com um vetor, é possível afirmar que:
   1. Na implementação de uma pilha, precisamos sempre de duas variáveis distintas, uma para apontar para o local onde ocorrem as inclusões e outra para apontar para o local onde ocorrem as exclusões.
   2. **O comportamento de uma pilha é semelhante ao comportamento do botão "desfazer" de editores de texto. Ao acionar o desfazer, o último comando inserido é o primeiro a ser desfeito.**
   3. Para implementar uma pilha usando vetor, é obrigatório que a alocação do vetor seja dinâmica, pois precisamos fazer com que o vetor aumente e diminua de tamanho em tempo de execução.
   4. Nas implementações de pilhas usando vetor, as inserções ocorrem em tempo constante, enquanto que as remoções dependem do tamanho do vetor.
   5. Nas implementações de pilha usando vetor, a variável que indica o topo da pilha precisa ser uma variável do tipo ponteiro para inteiro, dado que um vetor é, na verdade, um ponteiro para uma região de memória.

A resposta correta é: “O comportamento de uma pilha é semelhante ao comportamento do botão "desfazer" de editores de texto. Ao acionar o desfazer, o último comando inserido é o primeiro a ser desfeito."  
  
JUSTIFICATIVA  
  
Em “Incremental, Cascata, Recursivo e Imperativo”: Recursivo não é um tipo de ciclo de vida, é um método de implementação.  
Em “Funcional, Imperativo, Interativo e Orientado a Objetos.”: são paradigmas de desenvolvimento de Na implementação de uma pilha, precisamos sempre de duas variáveis distintas, uma para apontar para o local onde ocorrem as inclusões e outra para apontar para o local onde ocorrem as exclusões.  
Incorreto, as inclusões e exclusões ocorrem todas no topo da pilha, então basta uma variável para fazer a indicação de onde ocorrem.  
Para implementar uma pilha usando vetor, é obrigatório que a alocação do vetor seja dinâmica, pois precisamos fazer com que o vetor aumente e diminua de tamanho em tempo de execução.  
Incorreto, não é obrigatório que a alocação seja dinâmica. Além disso, mesmo com alocação dinâmica não podemos aumentar e diminuir o tamanho do vetor.  
O comportamento de uma pilha é semelhante ao comportamento do botão "desfazer" de editores de texto. Ao acionar o desfazer, o último comando inserido é o primeiro a ser desfeito.  
Correto  
Nas implementações de pilhas usando vetor, as inserções ocorrem em tempo constante, enquanto que as remoções dependem do tamanho do vetor.  
Incorreto, tanto as inserções quanto as remoções devem ocorrer em tempo constante.  
Nas implementações de pilha usando vetor, a variável que indica o topo da pilha precisa ser uma variável do tipo ponteiro para inteiro, dado que um vetor é, na verdade, um ponteiro para uma região de memória.  
Incorreto, a variável precisa ser do tipo inteiro, dado que o apontamento é feito usando o índice do vetor.

1. **(1 ponto)**Sobre a estrutura de dados fila com implementação com um vetor, é possível afirmar que:
   1. Uma fila é uma estrutura de dados tipicamente usada para resolver problemas de alinhamento de estruturas. Por exemplo, em linguagens de programação, garantir que a abertura dos escopos está sendo adequadamente finalizada.
   2. **Filas são estruturas que garantem acesso justo a recursos compartilhados. Isso ocorre porque o primeiro elemento a entrar na fila (o que pode ser interpretado como pedir acesso a um recurso) será o primeiro a sair (ser atendido).**
   3. Uma fila é uma estrutura dita não linear, dado que não é facilmente interpretada como um vetor. Por conta disso, implementações de fila usando vetores não são adequadas.
   4. Na implementação de uma fila usando um vetor, utilizamos uma variável para guardar a posição da cabeça da fila, sendo isso suficiente para implementar as operações de inserção e remoção, dado que ambas as operações ocorrem na cabeça da fila.
   5. Um grande problema da implementação de filas com vetores é que inserções ocupam espaço no final do vetor e remoções liberam espaço no início do vetor, o que impede que o espaço liberado seja reaproveitado. Assim, a fila pode ficar cheia mesmo tendo muito espaço livre no início do vetor, um desperdício comum às implementações de fila com vetor.

A resposta falsa é: “Filas são estruturas que garantem acesso justo a recursos compartilhados. Isso ocorre porque o primeiro elemento a entrar na fila (o que pode ser interpretado como pedir acesso a um recurso) será o primeiro a sair (ser atendido)."  
  
JUSTIFICATIVA  
  
Uma fila é uma estrutura de dados tipicamente usada para resolver problemas de alinhamento de estruturas. Por exemplo, em linguagens de programação, garantir que a abertura dos escopos está sendo adequadamente finalizada.  
Incorreto, a alternativa confunde fila com pilha.  
Uma fila é uma estrutura dita não linear, dado que não é facilmente interpretada como um vetor. Por conta disso, implementações de fila usando vetores não são adequadas.  
Incorreto, uma fila é uma estrutura dita linear.  
Filas são estruturas que garantem acesso justo a recursos compartilhados. Isso ocorre porque o primeiro elemento a entrar na fila (o que pode ser interpretado como pedir acesso a um recurso) será o primeiro a sair (ser atendido).  
Correto  
Na implementação de uma fila usando um vetor, utilizamos uma variável para guardar a posição da cabeça da fila, sendo isso suficiente para implementar as operações de inserção e remoção, dado que ambas as operações ocorrem na cabeça da fila.  
Incorreto, as operações de remoções e inserções ocorrem em pontos diferentes da fila.  
Um grande problema da implementação de filas com vetores é que inserções ocupam espaço no final do vetor e remoções liberam espaço no início do vetor, o que impede que o espaço liberado seja reaproveitado. Assim, a fila pode ficar cheia mesmo tendo muito espaço livre no início do vetor, um desperdício comum às implementações de fila com vetor.  
Incorreto. Este cenário da alternativa foi apresentado durante a aula. Entretanto, durante a própria aula foi criada uma solução para ele, que consiste em usar o resto da divisão pelo tamanho do vetor para encontrar a posição real do elemento no vetor. Fazendo, o vetor passa a ser imaginado como uma estrutura circular, ao invés de ser imaginado como uma estrutura linear.

1. **(1 ponto)**Sobre listas sequenciais e listas encadeadas, é correto afirmar que:
   1. Um vetor alocado estaticamente pode ser chamado de lista linear sequencial, pois a ordem lógica é igual à ordem física. O mesmo não ocorre com vetores alocados dinamicamente, pois estes podem ter seus elementos em lugares diversos na memória.
   2. **Para a busca binária funcionar com desempenho em tempo O(log n), precisamos da propriedade de que cada elemento possa ser acessado em tempo constante dado o índice. Nesse caso, é requerida uma lista linear sequencial, não sendo interessante usar lista encadeada.**
   3. Quando temos uma lista encadeada, o acesso a cada elemento, dado o índice, ocorre em tempo constante.
   4. Em uma lista encadeada, a ordem lógica dos elementos não corresponde ao posicionamento destes em memória. Por conta disso, é possível fazer acesso a qualquer elemento da lista em tempo constante.
   5. Em um vetor alocado dinamicamente, podemos aumentar ou diminuir o número de elementos. Nesse caso, é mais fácil alocar vetores com "new" do que estaticamente.

A resposta correta é: “Para a busca binária funcionar com desempenho em tempo O(log n), precisamos da propriedade de que cada elemento possa ser acessado em tempo constante dado o índice. Nesse caso, é requerida uma lista linear sequencial, não sendo interessante usar lista encadeada."  
  
JUSTIFICATIVA  
  
Um vetor alocado estaticamente pode ser chamado de lista linear sequencial, pois a ordem lógica é igual à ordem física. O mesmo não ocorre com vetores alocados dinamicamente, pois estes podem ter seus elementos em lugares diversos na memória.  
Incorreto, vetores alocados dinamicamente também são sequenciais.  
Quando temos uma lista encadeada, o acesso a cada elemento, dado o índice, ocorre em tempo constante.  
Incorreto, o acesso a cada elemento precisa seguir pelos elementos predecessores, não ocorrendo em tempo constante.  
Em uma lista encadeada, a ordem lógica dos elementos não corresponde ao posicionamento destes em memória. Por conta disso, é possível fazer acesso a qualquer elemento da lista em tempo constante.  
Incorreto, não é possível fazer acesso a qualquer elemento da lista em tempo constante.  
Para a busca binária funcionar com desempenho em tempo O(log n), precisamos da propriedade de que cada elemento possa ser acessado em tempo constante dado o índice. Nesse caso, é requerida uma lista linear sequencial, não sendo interessante usar lista encadeada.  
Correto.  
Em um vetor alocado dinamicamente, podemos aumentar ou diminuir o número de elementos. Nesse caso, é mais fácil alocar vetores com "new" do que estaticamente.  
Incorreto, não podemos aumentar ou diminuir o número de elementos em vetores alocados dinamicamente.

1. **(1 ponto)**Sobre as estruturas de dados pilha e fila, implementadas tanto com vetores quanto com listas encadeadas, é correto afirmar:
   1. Uma pilha implementada com vetores possui operações de push e pop em tempo constante. No caso da implementação com listas encadeadas, a operação de push possui tempo de execução determinado pelo número de elementos.
   2. **Tanto pilha ou fila implementada com vetores, quanto pilha ou fila implementada com listas encadeadas possuem operações de inserção e remoção de elementos em tempo constante.**
   3. Uma fila implementada com vetores possui operações de enqueue e dequeue em tempo constante. No caso da implementação com listas encadeadas, a operação de enqueue possui tempo de execução determinado pelo número de elementos.
   4. Uma pilha implementada com vetores possui operações de push e pop em tempo constante. No caso da implementação com listas encadeadas, a operação de pop possui tempo de execução determinado pelo número de elementos.
   5. Uma fila implementada com vetores possui operações de enqueue e dequeue em tempo constante. No caso da implementação com listas encadeadas, a operação de dequeue possui tempo de execução determinado pelo número de elementos.

A resposta correta é: “Tanto pilha ou fila implementada com vetores, quanto pilha ou fila implementada com listas encadeadas possuem operações de inserção e remoção de elementos em tempo constante."  
  
JUSTIFICATIVA  
  
Com a implementação de pilha em vetor, as operações de inserção e remoção simplesmente fazem operações locais no elemento marcado como topo. Isso envolve inserir ou remover um elemento do vetor e alterar o valor de uma variável do tipo inteiro, sendo que tudo pode ser feito independente do tamanho da pilha. Com a implementação em listas encadeadas, as operações de inserção e remoção envolvem alocar ou desalocar memória, respectivamente, seguida da atualização de um número constante de variáveis do tipo ponteiro.  
No caso de fila com a implementação em vetor, as operações de inserção e remoção envolvem inserir ou remover um único elemento que pode ser acessado em tempo constante com o índice, seguida da atualização dos valores de uma variável, que pode ser a que guarda o índice de quem está na frente, ou a que guarda o índice de quem está atrás, o que implica dizer que tudo ocorre independentemente do tamanho da fila. Com a implementação em listas encadeadas, as operações de inserção e remoção envolvem (do mesmo modo como ocorreu na pilha) alocar ou desalocar memória, respectivamente, seguida da atualização de um número constante de variáveis do tipo ponteiro.

1. **(1 ponto)**O código a seguir implementa um método em uma dada estrutura de dados, que pode ser uma pilha, implementada em aula na classe Stack, ou uma fila, implementada em aula na classe Queue. No código, a variável "structure" é um ponteiro para o início da estrutura de dados.

Sobre ele, é possível afirmar que:

* 1. O código implementa a função push da classe Stack, sendo que a estrutura interna é um vetor.
  2. **O código implementa a função push da classe Stack, sendo que a estrutura interna é uma lista encadeada.**
  3. O código implementa a função enqueue da classe Queue, sendo que a estrutura interna é um vetor.
  4. O código implementa a função enqueue da classe Queue, sendo que a estrutura interna é uma lista encadeada.
  5. O código implementa a função dequeue da classe Queue, sendo que a estrutura interna é uma lista encadeada.

A resposta correta é: “O código implementa a função push da classe Stack, sendo que a estrutura interna é uma lista encadeada."  
  
JUSTIFICATIVA  
  
O código possui alocação dinâmica no método de inserção de elementos, o que descarta o uso de um vetor como implementação interna. Além disso, o código está alocando o novo nó logo no início da estrutura com a linha "structure = location", algo que não seria feito se a estrutura de dados fosse. Lembrando que, se a estrutura de dados fosse uma fila, a inserção exigiria que um elemento que já está na estrutura apontasse para o elemento sendo inserido, algo que não vemos no trecho de código.

ATIVIDADE PARA AVALIAÇÃO - SEMANA 4

A resposta correta da questão está identificada com a cor Vermelha.

1. **(1 ponto)**Sobre Tabelas Hash, é possível afirmar que:
   1. Uma Tabela Hash difere dos vetores porque na Tabela Hash temos índices do tipo string e valores do tipo inteiro, enquanto nos vetores temos índices do tipo inteiro e valores do tipo string.
   2. **Se internamente implementamos uma Tabela Hash como um vetor, precisaremos de uma função de mapeamento que mapeie as chaves para inteiros, dentro dos limites estabelecidos pelo vetor.**
   3. Em uma Tabela Hash, cada entrada é do tipo (k,v), onde a chave k precisa ser do tipo inteiro e v pode receber qualquer tipo.
   4. A implementação de uma Tabela Hash faz com que as buscas ocorram em tempo constante. As deleções e inserções ocorrem em tempo O(log n).
   5. Em uma Tabela Hash implementada como vetor, o tamanho do vetor não costuma interferir na presença de colisões, a não ser que algum erro de projeto tenha sido cometido pelos desenvolvedores.

A resposta correta é: “Se internamente implementamos uma Tabela Hash como um vetor, precisaremos de uma função de mapeamento que mapeie as chaves para inteiros, dentro dos limites estabelecidos pelo vetor.”  
  
JUSTIFICATIVA  
  
Uma Tabela Hash difere dos vetores porque na Tabela Hash temos índices do tipo string e valores do tipo inteiro, enquanto nos vetores temos índices do tipo inteiro e valores do tipo string.  
Incorreto, não existe limitação para os valores adicionados na Tabela Hash e nem para os valores adicionados no vetor. Assim como não há limitação para as chaves da Tabela Hash.  
Em uma Tabela Hash, cada entrada é do tipo (k,v), onde a chave k precisa ser do tipo inteiro e v pode receber qualquer tipo.  
Incorreto, a chave k não precisa ser do tipo inteiro.  
Se internamente implementamos uma Tabela Hash como um vetor, precisaremos de uma função de mapeamento que mapeie as chaves para inteiros, dentro dos limites estabelecidos pelo vetor.  
Correto.  
A implementação de uma Tabela Hash faz com que as buscas ocorram em tempo constante. As deleções e inserções ocorrem em tempo O(log n).  
Incorreto, as deleções e inserções também ocorreriam em tempo constante.  
Em uma Tabela Hash implementada como vetor, o tamanho do vetor não costuma interferir na presença de colisões, a não ser que algum erro de projeto tenha sido cometido pelos desenvolvedores.  
Incorreto, o tamanho do vetor é um dos parâmetros que interferem na ocorrência de colisões.

1. **(1 ponto)**Uma função de mapeamento de strings para inteiros pontua as vogais como: a = 1, e = 3, i = 5, o = 7, u = 9. Todos os outros caracteres são pontuados com 0 (zero). Um inteiro que representa a string é computado com a soma das pontuações dos caracteres. Quantas colisões ocorrem se tivermos as palavras: ulisses, arthur, danielle, christina, paula.
   1. 0
   2. 2
   3. 3
   4. 4
   5. **1**

A resposta correta é: “1”  
  
JUSTIFICATIVA  
  
Para resolver esta questão, é necessário computar o valor de todas as palavras para saber se colisões ocorrerão.  
  
ulisses → vogais: u, i, e → 9 + 5 + 3 → 17  
arthur → vogais: a, u → 1 + 9 → 10  
danielle → vogais: a, i, e, e → 1 + 5 + 3 + 3 → 12  
christina → vogais: i, i, a → 5 + 5 + 1 → 11  
paula → vogais: a, u, a → 1 + 9 + 1 → 11  
  
As únicas colisões que ocorrem são entre as palavras christina e paula, ambas mapeando para 11.

1. **(1 ponto)**Sobre Função de Hash, é possível afirmar que:
   1. Uma boa função de hash é aquela que minimiza o espaço de memória utilizado, alocando apenas o número de endereços realmente utilizados.
   2. Uma boa função de hash é aquela que utiliza uma implementação interna com um vetor cujo tamanho é um número primo, independentemente do número de colisões.
   3. **Uma boa função de hash é aquela que diminui o número de colisões, o que permite inserções, remoções e atualizações de maneira eficiente.**
   4. Uma boa função de hash é aquela que garante que todas as chaves serão mapeadas em números inteiros.
   5. Uma boa função de hash é aquela que fará com que os registros fiquem em posições contíguas de memória, o que facilitará a implementação da busca binária e conferirá eficiência para inserções, remoções e atualizações.

A resposta correta é: “Uma boa função de hash é aquela que diminui o número de colisões, o que permite inserções, remoções e atualizações de maneira eficiente.”  
  
JUSTIFICATIVA  
  
Na videoaula, uma função de hash foi caracterizada como boa se minimizar a quantidade de colisões. Dito isso, a classificação como boa deve ser feita para qualquer problema, exigindo um estudo do número esperado de colisões para verificar se ele é menor do que uma dada constante. Dessa forma, a definição não está vinculada a detalhes de implementação, como uso de um vetor ou mapeamento para números inteiros.

1. **(1 ponto)**Uma função de hash para mapeamento de strings em inteiros usa a tabela ascii para mapear cada caractere da string em um número. Seja C= (c0,c1,c2,...,ck-1) os valores numéricos dos caracteres nas posições 0,1, ... k-1, a função é dada por.

Onde a é uma constante qualquer e N é o número máximo de elementos na tabela.

Sobre essa função, é correto afirmar que.

* 1. **A função tenta evitar colisões entre strings s1 e s2 que são anagramas uma da outra. Isso permite distribuir mais adequadamente as strings na tabela hash se comparada com uma função que não leva em conta as posições dos caracteres.**
  2. Se uma string s for um palíndromo (mesma leitura da direita para esquerda e da esquerda para direita), então teremos uma colisão entre s e outra string que possui os mesmos caracteres de s.
  3. Se duas strings s1 e s2 possuem os mesmos caracteres, então teremos uma colisão.
  4. A função é isenta de colisões no que diz respeito a strings que possuem os mesmos caracteres em qualquer ordem, independente de quem são as strings e do valor de N.
  5. As palavras adriana e ariadna serão mapeadas para posições diferentes, mas orlando e odnalro serão mapeados para a mesma posição.

A resposta correta é: “A função tenta evitar colisões entre strings s1 e s2 que são anagramas uma da outra. Isso permite distribuir mais adequadamente as strings na tabela hash se comparada com uma função que não leva em conta as posições dos caracteres. ”  
  
JUSTIFICATIVA  
  
A função de hash foi apresentada em um formato genérico, dado que não sabemos os valores de a e de N. Nesse caso, não é possível afirmar de uma maneira genérica que strings palíndromos e anagramas da mesma palavra gerarão colisões ou não. Se o N escolhido for 2, por exemplo, metade das palavras gerarão colisão, podendo essas colisões virem de anagramas ou não. A única afirmação que podemos fazer é que a função tenta espalhar as chaves de uma maneira melhor do que uma outra função que não leva o critério da posição dos caracteres em conta.

1. **(1 ponto)**Um código em C++ da função retrieveItem de uma implementação de tabela hash é mostrado a seguir. Este código busca um elemento no vetor "structure" usando uma chave de busca presente na variável "aluno" recebida como parâmetro. Esse mesmo parâmetro "aluno" é usado para retornar o conteúdo da tabela Hash.

Analisando esse código, assinale a alternativa correta.

* 1. A função trata todas as colisões, exceto aquelas formadas por strings s1 e s2 que são anagramas uma da outra.
  2. **A função funcionaria se assumirmos um cenário ideal em que as colisões não existem. Nesse caso, precisa ainda ser adaptada se quisermos que funcione em cenários com colisões.**
  3. A função assume um cenário em que as colisões são tratadas com teste linear. Nesse caso, funcionaria sem erros na presença de colisões, apesar de ainda ser possível fazer algumas melhorias.
  4. A função assume que strings não são palíndromos, dado que deixaria de funcionar nesses casos.
  5. A função assume que as chaves são números reais que podem ser mapeados em números inteiros.

A resposta correta é: “A função funcionaria se assumirmos um cenário ideal em que as colisões não existem. Nesse caso, precisa ainda ser adaptada se quisermos que funcione em cenários com colisões."  
  
JUSTIFICATIVA  
  
A função apresentada é uma implementação básica dos conceitos para cenários sem colisões, dado que busca apenas na posição do vetor dada pela função de hash. Em caso de colisões, uma segunda busca deveria ter sido implementada para encontrar o registro em outra posição de memória caso na posição indicada pela função de hash exista um elemento com chave diferente, algo que não foi feito na função.

1. **(1 ponto)**Um código em C++ das funções insertItem e deleteItem de uma implementação de tabela hash é mostrado a seguir. A função insertItem insere o objeto "aluno" que veio por parâmetro e a função deleteItem remove um registro na posição indicada pela função de hash aplicada ao objeto "aluno" que veio por parâmetro.

Analisando esse código, assinale a alternativa correta.

* 1. Em caso de colisão, a função insertItem não altera o valor que está na tabela, mantendo o valor antigo.
  2. **Em caso de colisão, a função deleteItem pode apagar um aluno com chave diferente daquele passado por parâmetro.**
  3. Em um cenário ideal de ausência completa de colisões, a função deleteItem funcionaria corretamente, mas insertItem apresentaria erros em tempo de compilação.
  4. Em um cenário ideal de ausência completa de colisões, a função insertItem funcionaria corretamente, mas deleteItem apresentaria erros em tempo de compilação.
  5. As funções tratam colisões por estarem usando a técnica de teste linear.

A resposta correta é: “Em caso de colisão, a função deleteItem pode apagar um aluno com chave diferente daquele passado por parâmetro."  
  
JUSTIFICATIVA  
  
Ambas as funções insertItem e deleteItem funcionariam em um cenário ideal de ausência completa de colisões, dado que são implementações básicas da teoria de Tabela Hash. Entretanto, geram problemas se colisões ocorrerem. Por exemplo, insertItem irá substituir alguma chave existente na tabela caso receba um registro com chave diferente, mas apontando para o mesmo endereço. Por sua vez, deleteItem poderá apagar um registro diferente daquele que veio por parâmetro se a função de hash indicar a mesma posição no vetor.

1. **(1 ponto)**Sobre tratamento de colisões, é correto afirmar que:
   1. Tratar colisões significa fazer mudanças na função de hash de modo a diminuir as chances de que elas ocorram.
   2. **Tratar colisões significa que, no caso de ocorrência de colisões, um dos registros ficará em um segundo endereço de memória, que também será analisado em caso de busca, atualização e remoção do registro.**
   3. Tratar colisões significa alocar um espaço fora do vetor para alocar em um buffer elementos com chaves que colidiram. Quando liberar espaço no vetor, os elementos serão movidos do buffer de volta para o vetor.
   4. Tratar colisões significa manter uma média de X/Y entradas por endereço, sendo X o número de elementos e Y o tamanho do vetor.
   5. Tratar colisões significa implementar políticas que garantem que elas não ocorrerão.

A resposta falsa é: “Tratar colisões significa que, no caso de ocorrência de colisões, um dos registros ficará em um segundo endereço de memória, que também será analisado em caso de busca, atualização e remoção do registro."  
  
JUSTIFICATIVA  
  
Tratar colisões implica dizer que os registros não serão perdidos, mesmo que a função de hash mapeie duas chaves diferentes para a mesma região de memória. Nesse caso, um dos registros deverá ficar alocado em uma segunda posição de memória e esta posição será também buscada nas operações de consulta, inserção e remoção.

1. **(1 ponto)**Duas ideias para tratamento de colisões são o encadeamento separado e o teste linear. Sobre elas, podemos afirmar que:
   1. O encadeamento separado é melhor que o teste linear por não exigir que a função de mapeamento distribua muito bem os dados. Será possível fazer buscas em tempo constante mesmo que a função mapeie todos os registros para o mesmo endereço.
   2. **O encadeamento separado utiliza uma região de memória adicional a ser implementada possivelmente com uma lista encadeada para tratar colisões. O teste linear utiliza a região de memória da própria tabela para colocar os registros que colidiram.**
   3. O teste linear é melhor que o encadeamento separdo por não exigir que a função de mapeamento distribua muito bem os dados. Será possível fazer buscas em tempo constante mesmo que a função mapeie todos os registros para o mesmo endereço.
   4. O teste linear utiliza uma região de memória adicional a ser implementada possivelmente com uma lista encadeada para tratar colisões. O encadeamento separado utiliza a região de memória da própria tabela para colocar os registros que colidiram.
   5. O teste linear tende a aumentar a complexidade assintótica do algoritmo no pior caso para O(log n). Isso ocorre mesmo quando a função de hash distribui os dados uniformemente.

A resposta correta é: “O encadeamento separado utiliza uma região de memória adicional a ser implementada possivelmente com uma lista encadeada para tratar colisões. O teste linear utiliza a região de memória da própria tabela para colocar os registros que colidiram."  
  
JUSTIFICATIVA  
  
O encadeamento separado é melhor que o teste linear por não exigir que a função de mapeamento distribua muito bem os dados. Será possível fazer buscas em tempo constante mesmo que a função mapeie todos os registros para o mesmo endereço.  
Incorreto, se a função mapear todos os registros para o mesmo lugar, então não poderemos fazer buscas em tempo constante.  
O teste linear é melhor que o encadeamento separado por não exigir que a função de mapeamento distribua muito bem os dados. Será possível fazer buscas em tempo constante mesmo que a função mapeie todos os registros para o mesmo endereço.  
Incorreto, se a função mapear todos os registros para o mesmo lugar, então não poderemos fazer buscas em tempo constante, independente de usarmos teste linear ou encadeamento separado.  
O encadeamento separado utiliza uma região de memória adicional a ser implementada possivelmente com uma lista encadeada para tratar colisões. O teste linear utiliza a região de memória da própria tabela para colocar os registros que colidiram.  
Correto.  
O teste linear utiliza uma região de memória adicional a ser implementada possivelmente com uma lista encadeada para tratar colisões. O encadeamento separado utiliza a região de memória da própria tabela para colocar os registros que colidiram.  
Incorreto, a alternativa troca teste linear e encadeamento separado.  
O teste linear tende a aumentar a complexidade assintótica do algoritmo no pior caso para O(log n). Isso ocorre mesmo quando a função de hash distribui os dados uniformemente.  
Incorreto, a complexidade assintótica O(log n) não está correta neste contexto, dado que sequer é estabelecido como a mesma pode ser obtida em uma tabela Hash. Lembrando que neste contexto não cabe o uso de busca binária, que poderia induzir a complexidade O(log n).

1. **(1 ponto)**Sobre o teste linear para tratamento de colisão em uma tabela hash, assinale a alternativa correta sabendo que posições na implementação interna do vetor podem estar "ocupadas", "vazias" ou "disponíveis".
   1. A função de inserção tenta colocar um novo registro na posição indicada pela função de hash. Caso não consiga, o elemento é colocado em um espaço de memória adicional fora do vetor.
   2. **A função de busca de um elemento procura a chave de entrada, inicialmente na posição indicada pela função de hash, mas, caso ela esteja ocupada por outra entrada, inicia uma busca sequencial nas posições seguintes e só pára quando encontrar o elemento, ou quando encontrar uma posição "vazia".**
   3. A função de remoção procura a entrada a ser apagada na posição indicada pela função de hash. Caso esta posição esteja ocupada por uma outra entrada com chave diferente, inicia uma busca sequencial nas posições seguintes e só pára quando encontrar o elemento a ser apagado, ou quando encontrar uma posição "disponível".
   4. A função de inserção tenta colocar um novo registro na posição indicada pela função de hash. Caso a posição esteja ocupada, uma busca inicia nas posições seguintes até encontrar uma posição marcada como "disponível".
   5. A função de inserção tenta colocar um novo registro na posição indicada pela função de hash. Caso a posição esteja ocupada, uma busca inicia nas posições seguintes até encontrar uma posição marcada como "vazia".

A resposta correta é: “A função de busca de um elemento procura a chave de entrada, inicialmente na posição indicada pela função de hash, mas, caso ela esteja ocupada por outra entrada, inicia uma busca sequencial nas posições seguintes e só pára quando encontrar o elemento, ou quando encontrar uma posição "vazia"."  
  
JUSTIFICATIVA  
  
A função de busca de um elemento procura a chave de entrada, inicialmente na posição indicada pela função de hash, mas, caso ela esteja ocupada por outra entrada, inicia uma busca sequencial nas posições seguintes e só páara quando encontrar o elemento, ou quando encontrar uma posição "vazia"  
Correto.  
A função de inserção tenta colocar um novo registro na posição indicada pela função de hash. Caso não consiga, o elemento é colocado em um espaço de memória adicional fora do vetor.  
Incorreto, o teste linear não utiliza espaço adicional, apenas o espaço do vetor.  
A função de remoção procura a entrada a ser apagada na posição indicada pela função de hash. Caso esta posição esteja ocupada por uma outra entrada com chave diferente, inicia uma busca sequencial nas posições seguintes e só para quando encontrar o elemento a ser apagado, ou quando encontrar uma posição disponível.  
Incorreto, a busca pararia se o elemento a ser apagado fosse encontrado, ou se fosse alcançada uma posição vazia.  
A função de inserção tenta colocar um novo registro na posição indicada pela função de hash. Caso a posição esteja ocupada, uma busca inicia nas posições seguintes até encontrar uma posição marcada como "disponível".  
Incorreto, a inserção procura tanto por uma posição "vazia" quanto por uma posição "disponível". A função de inserção tenta colocar um novo registro na posição indicada pela função de hash. Caso a posição esteja ocupada, uma busca inicia nas posições seguintes até encontrar uma posição marcada como "vazia".  
Incorreto, a inserção procura tanto por uma posição "vazia" quanto por uma posição "disponível".

1. **(1 ponto)**Sobre o teste linear para tratamento de colisão em uma tabela hash, assinale a alternativa correta sabendo que posições na implementação interna do vetor podem estar "ocupadas", "vazias" ou "disponíveis".
   1. A função de remoção de uma entrada deverá procurar um elemento com chave igual ao que foi recebido por parâmetro. A busca deve ser feita usando a busca binária, iniciando na posição indicada pela função de hash./li>
   2. **A função de remoção de uma entrada deverá procurar um elemento com chave igual ao que foi recebido por parâmetro. Encontrado o elemento, marca a posição como "disponível".**
   3. A função de remoção remove o elemento que está na posição indicada pela função de hash. Isso ocorre porque as colisões são um problema a ser tratado apenas nas buscas e inserções.
   4. A função de remoção, após encontrar o elemento a ser removido, marca a posição onde ele está como "vazia".
   5. A função de remoção deve ser chamada com elementos que realmente existem na tabela, senão entrará em looping infinito.

A resposta correta é: “A função de remoção de uma entrada deverá procurar um elemento com chave igual ao que foi recebido por parâmetro. Encontrado o elemento, marca a posição como "disponível"."  
  
JUSTIFICATIVA  
  
A função de remoção de uma entrada deverá procurar um elemento com chave igual ao que foi recebido por parâmetro. Encontrado o elemento, marca a posição como "disponível".  
Correto.  
A função de remoção de uma entrada deverá procurar um elemento com chave igual ao que foi recebido por parâmetro. A busca deve ser feita usando a busca binária, iniciando na posição indicada pela função de hash.  
Incorreto, a ideia de busca binária não é aplicável no contexto.  
A função de remoção remove o elemento que está na posição indicada pela função de hash. Isso ocorre porque as colisões são um problema a ser tratado apenas nas buscas e inserções.  
Incorreto, colisões obrigam a troca das implementações de inserções, remoções e exclusões.  
A função de remoção, após encontrar o elemento a ser removido, marca a posição onde ele está como "vazia".  
Incorreto, a posição é marcada como "disponível".  
  
  
Descontar 20% do valor da questão por cada posicionamento errado.

ATIVIDADE PARA AVALIAÇÃO - SEMANA 5

A resposta correta da questão está identificada com a cor Vermelha.

1. **(1 ponto)**Sobre árvores, assinale a alternativa correta.
   1. Em uma árvore, o grau de um nó é o número de arestas ligadas a esse nó. Nesse caso, um nó que possui um pai e dois filhos possui grau 3.
   2. **A profundidade de um nó pode ser computada pela distância deste nó até a raiz.**
   3. A profundidade de um nó é a distância deste nó até o nó descendente mais distante.
   4. A altura de uma árvore é a distância entre a raiz e a folha mais próxima.
   5. A altura de um nó é a distância deste nó até a raiz.

A resposta correta é: “A profundidade de um nó pode ser computada pela distância deste nó até a raiz.”  
  
JUSTIFICATIVA  
  
Em uma árvore, o grau de um nó é o número de arestas ligadas a esse nó. Nesse caso, um nó que possui um pai e dois filhos possui grau 3. Incorreto, o grau de um nó é o número de subárvores ligadas a ele.  
Profundidade de um nó pode ser computada pela distância desse nó até a raiz. Correto.  
A profundidade de um nó é a distância deste nó até o nós descendente mais distante. Incorreto, a profundidade é computada em relação à raiz da árvore.  
A altura de uma árvore é a distância entre a raiz e a folha mais próxima. Incorreto, seria em relação à folha mais distante.  
A altura de um nó é a distância deste nó até a raiz. Incorreto, seria em relação à folha mais distante.

1. **(1 ponto)**Sobre árvores binárias, assinale a alternativa correta.
   1. Em uma árvore binária, o nó raiz sempre possui grau 2.
   2. Em uma árvore binária, o grau máximo de um nó é 3, o que ocorre quando uma árvore possui um pai e duas subárvores.
   3. Em uma árvore binária, as chaves do lado esquerdo são menores que as chaves do lado direito.
   4. Em uma árvore binária, as chaves das subárvores filhas são menores que as chaves do nó.
   5. **Em uma árvore binária, as folhas possuem nós de grau zero e os nós internos possuem graus de no máximo 2.**

A resposta correta é: “Em uma árvore binária, as folhas possuem nós de grau zero e os nós internos possuem graus de no máximo 2.”  
  
JUSTIFICATIVA  
  
Em uma árvore binária, o nó raiz sempre possui grau 2. Incorreto, a raiz pode ter grau 0, 1 ou 2.  
Em uma árvore binária, o grau máximo de um nó é 3, o que ocorre quando uma árvore possui um pai e duas subárvores. Incorreto, o grau máximo de um nó é 2, pois o grau é a quantidade de nós filhos.  
Em uma árvore binária, as folhas possuem nós de grau zero e os nós internos possuem graus de no máximo 2. Correto.  
Em uma árvore binária, as chaves do lado esquerdo são menores que as chaves do lado direito. Incorreto, estamos falando de árvore binária, não estamos falando de uma árvore binária de busca.  
Em uma árvore binária, as chaves das subárvores filhas são menores que as chaves do nó. Incorreto, estamos falando de árvore binária, não estamos falando de Heap.

1. **(1 ponto)**Dada a árvore binária a seguir, assinale a alternativa correta.
   1. A altura de 26 é 1.
   2. A altura de 18 é 4.
   3. **A profundidade do nó 81 é 3.**
   4. O grau de 19 é 2.
   5. A profundidade de 70 é 1.

A resposta correta é: “Uma boa função de hash é aquela que diminui o número de colisões, o que permite inserções, remoções e atualizações de maneira eficiente.”  
  
JUSTIFICATIVA  
  
A profundidade do nó 81 é 3. Correto.  
A altura de 26 é 1. Incorreto, a altura de 26 é 2.  
A altura de 18 é 4. Incorreto, a altura de 18 é 3.  
O grau de 19 é 2. Incorreto, o grau de 19 é 1.  
A profundidade de 70 é 1. Incorreto, a profundidade de 70 é 2.

1. **(ANULADA)**Sobre árvores binárias de busca, assinale a alternativa correta.
   1. **Ao inserir um nó cuja chave não existe na árvore, colocamos o novo nó na posição em que ele estaria se fosse buscado. Assim, o novo nó é sempre inserido como nó folha.**
   2. Inserções, remoções e buscas possuem complexidade de tempo proporcional ao número de nós em árvores de busca balanceadas.
   3. As chaves dos nós filhos são maiores que as chaves dos nós pais.
   4. Os filhos da esquerda são percorridos antes dos filhos da direita na busca por elementos dada uma chave qualquer.
   5. Inserções em árvores binárias de busca aumentam a altura em +1.

A resposta correta é: “Ao inserir um nó cuja chave não existe na árvore, colocamos o novo nó na posição em que ele estaria se fosse buscado. Assim, o novo nó é sempre inserido como nó folha. ”  
  
JUSTIFICATIVA  
  
Inserções, remoções e buscas possuem complexidade de tempo proporcional ao número de nós em árvores de busca balanceadas. Incorreto, em árvores balanceadas as operações ocorrem em O(log n).  
As chaves dos nós filhos são maiores que as chaves dos nós pais. Incorreto, a alternativa confunde árvore binária de busca com uma fila de prioridade, que não foi abordada na disciplina.  
Os filhos da esquerda são percorridos antes dos filhos da direita na busca por elementos dada uma chave qualquer. Incorreto, a chave é quem dita a busca, informando se vamos seguir para a esquerda ou para a direita.  
Ao inserir um nó cuja chave não existe na árvore, colocamos o novo nó na posição em que ele estaria se fosse buscado. Assim, o novo nó é sempre inserido como nó folha. Correto.  
Inserções em árvores binárias de busca aumentam a altura em +1. Incorreto, isso nem sempre é verdade.

1. **(1 ponto)**Sobre remoções em árvores binárias de busca, assinale a alternativa correta.
   1. Um nó só pode ser removido se for folha, caso contrário, mantemos o nó e marcamos como "disponível".
   2. **Se o nó removido possui apenas um filho, removemos o nó e colocamos o filho no lugar dele.**
   3. Se o nó a ser removido possui dois filhos, removemos o nó e colocamos o filho da esquerda no lugar.
   4. Se o nó a ser removido possui dois filhos, removemos o nó e colocamos o filho da direita no lugar.
   5. Se o nó a ser removido possui dois filhos, removemos o nó e colocamos o maior filho no lugar.

A resposta correta é: “Se o nó removido possui apenas um filho, removemos o nó e colocamos o filho no lugar dele."  
  
JUSTIFICATIVA  
  
Um nó só pode ser removido se for folha, caso contrário, mantemos o nó e marcamos como "disponível". Incorreto, a alternativa confunde com a abordagem feita na implementação de tabela hash com vetor.  
Se o nó removido possui apenas um filho, removemos o nó e colocamos o filho no lugar dele. Correto.  
Se o nó a ser removido possui dois filhos, removemos o nó e colocamos o filho da esquerda no lugar. Incorreto, podemos colocar o sucessor lógico ou o predecessor lógico.  
Se o nó a ser removido possui dois filhos, removemos o nó e colocamos o filho da direita no lugar. Incorreto, podemos colocar o sucessor lógico ou o predecessor lógico.  
Se o nó a ser removido possui dois filhos, removemos o nó e colocamos o maior filho no lugar. Incorreto, podemos colocar o sucessor lógico ou o predecessor lógico.

1. **(ANULADA)**Seja a árvore a seguir, qual alternativa mostra uma árvore obtida após a remoção de B usando o algoritmo aprendido em aula.

A resposta correta é: “"  
  
JUSTIFICATIVA  
  
No algoritmo visto em sala de aula, o nó que tem dois filhos vai ser substituído pelo sucessor lógico. Nesse caso, para identificar na árvore onde está o sucessor lógico, é preciso buscar o elemento mais à esquerda da subárvore da direita, o que seria o elemento G, que está à esquerda de D, raiz da subárvore da direita. Assim, na configuração final, o elemento G deve ficar no lugar onde antes estava o elemento B, removido da configuração final.

1. **(1 ponto)**Dados os percursos pré-ordem, pós-ordem e in-ordem a seguir, qual alternativa corretamente identifica a árvore em questão.

Pré-ordem: A C E F D G H

Pós-ordem: F E G H D C A

In-Ordem: E F C G D H A

A resposta correta é: “"  
  
JUSTIFICATIVA  
  
A alternativa B é que possui as leituras em pré-ordem, pós-ordem e in-ordem. Observe que a leitura em pré-ordem imprime primeiro o nó, depois a subárvore da esquerda e depois a subárvore da direita, resultando em A, C, E, F, D, G e H. No in-ordem, primeiro visitamos a subárvore da esquerda antes de visitar o nó, resultando em E, F, C, G, D, H e A. No pós-ordem, o nó só é visitado depois de passarmos pela subárvore da esquerda e da direita, o que gera F, E, G, H, D, C e A.

1. **(1 ponto)**Dada a árvore a seguir, qual alternativa identifica corretamente os percursos pré-ordem, pós-ordem e in-ordem, respectivamente?
   1. A B C D E F G H; C D B G F H E A; C D B A E F G H
   2. **A B C D E F G H; C D B G H F E A; C B D A G F H E**
   3. A B C D E F G H; C D B G H F E A; C D B A E F G H
   4. C D B A G H F E; C D B G H F E A; A B C D E F G H
   5. C D B A E F G H; C D B G H F E A; A B C D E F G H

A resposta correta é: “A B C D E F G H; C D B G H F E A; C B D A G F H E"  
  
JUSTIFICATIVA  
  
A leitura em pré-ordem imprime primeiro o nó, depois a subárvore da esquerda e depois a subárvore da direita, resultando em A, B, C, D, E, F, G e H. No in-ordem, primeiro visitamos a subárvore da esquerda antes de visitar o nó, resultando em C, B, D, A, G, F, H e E. No pós-ordem, o nó só é visitado depois de passarmos pela subárvore da esquerda e da direita, o que gera C, D, B, G, H, F, E e A.

1. **(ANULADA)**Em sala de aula, vimos o seguinte trecho de código usado em um dos casos da deleção de nó:

O método getSuccessor é mostrado a seguir:

Um aluno resolveu modificar os dois códigos da seguinte maneira. Mudou o primeiro trecho para invocar uma função diferente da getSuccessor que antes era invocada e trocou tree->direita para tree->esquerda, gerando:

O metodo getSomeNode foi implementado da seguinte maneira:

Com essas alterações, é correto afirmar que:

* 1. O código entrará em looping infinito, dado que não mais conseguiremos encontrar a chave procurada para substituir o nó que iremos remover.
  2. **A remoção de elementos continuará funcionando corretamente, dado que qualquer remoção gera um resultado que também é uma árvore binária de busca. Entretanto, a árvore binária de busca gerada quando o novo método for executado é diferente da árvore binária de busca gerada anteriormente.**
  3. A remoção de elementos continuará funcionando corretamente, dado que qualquer remoção gera a mesma árvore binária de busca que gerava anteriormente. As mudanças propostas não mudam o comportamento do código original.
  4. O código não entrará em looping infinito e realmente removerá o nó. Entretanto, a árvore binária gerada violará propriedades de árvore binária de busca no que diz respeito a ter os filhos menores do lado esquerdo e os filhos maiores do lado direito.
  5. A remoção de elementos não ocorrerá, dado que é possível pedir a remoção de um nó e ele não mais ser encontrado na árvore.

A resposta correta é: “A remoção de elementos continuará funcionando corretamente, dado que qualquer remoção gera um resultado que também é uma árvore binária de busca. Entretanto, a árvore binária de busca gerada quando o novo método for executado é diferente da árvore binária de busca gerada anteriormente.."  
  
JUSTIFICATIVA  
  
A remoção de elementos continuará funcionando corretamente, dado que qualquer remoção gera um resultado que também é uma árvore binária de busca. Entretanto, a árvore binária de busca gerada quando o novo método for executado é diferente da árvore binária de busca gerada automaticamente. Correto, o código está agora procurando o predecessor lógico ao invés do sucessor lógico para ocupar o lugar do nó que será removido.  
O código entrará em looping infinito, dado que não mais conseguiremos encontrar a chave procurada para substituir o nó que iremos remover. Incorreto, não haverá looping infinito nesta situação, pois conseguirá encontrar o nó a ser removido e atualizará a árvore de maneira adequada.  
A remoção de elementos continuará funcionando corretamente, dado que qualquer remoção gera a mesma árvore binária de busca que gerava anteriormente. As mudanças propostas não mudam o comportamento do código original. Incorreto, ao usar o predecessor lógico, a árvore gerada após a remoção será binária de busca, mas não será igual à anterior.  
O código não entrará em looping infinito e realmente removerá o nó. Entretanto, a árvore binária gerada violará propriedades de árvore binária de busca no que diz respeito a ter os filhos menores do lado esquerdo e os filhos maiores do lado direito. Incorreto, a árvore gerada será binária de busca.  
A remoção de elementos não ocorrerá, dado que é possível pedir a remoção de um nó e ele não mais ser encontrado na árvore. Incorreto, o nó pedido será removido.

1. **(1 ponto)**Em uma árvore binária de busca, foram inseridos os seguintes nós, nessa ordem: 5, 25, 6, 24, 7, 23, 8, 22, 9, 21, 10, 20, 11, 19, 12, 18, 13, 17, 14, 16, 15.  Sobre essa árvore, é possível afirmar que:
   1. No pior caso, a remoção de um nó demandará O(log n), sendo n o número de nós.
   2. **A árvore gerada não é balanceada. Nesse caso, as buscas ocorrerão com tempo proporcional ao número de nós na árvore.**
   3. No pior caso, a inserção de um nó demandará O(log n), sendo n o número de nós.
   4. A árvore gerada não será binária de busca, dado que existe um padrão na inserção dos nós.
   5. A árvore gerada não é balanceada. Entretanto, as buscas, inserções e remoções podem ser feitas de forma eficiente em O(log n).

A resposta correta é: “A árvore gerada não é balanceada. Nesse caso, as buscas ocorrerão com tempo proporcional ao número de nós na árvore."  
  
JUSTIFICATIVA  
  
A árvore gerada é:

Como podemos notar, temos uma árvore muito desbalanceada, que funciona basicamente como uma estrutura linear encadeada. Nesse caso, as inserções, remoções e buscas ocorrerão com tempo proporcional ao número de elementos, não podendo ser consideradas eficientes.

ATIVIDADE PARA AVALIAÇÃO - SEMANA 6

A resposta correta da questão está identificada com a cor Vermelha.

1. **(1 ponto)**Seja esq a subárvore esquerda de um nó e dir a subárvore direita dele em uma árvore AVL, é correto afirmar que:
   1. O fator de balanceamento do nó é dado pela subtração do número de nós em dir com o número de nós em esq.
   2. **O fator de balanceamento do nó é dado pela diferença de altura de dir e de esq.**
   3. A árvore será dita balanceada se dir tiver a mesma quantidade de nós que esq.
   4. O fator de balanceamento do nó é dado pela diferença de altura do nó em relação à soma das alturas das árvores esq e dir.
   5. As árvores esq e dir podem ter fator de balanceamento diferentes desde que tenham a mesma altura.

A resposta correta é: “O fator de balanceamento do nó é dado pela diferença de altura de dir e de esq.”  
  
JUSTIFICATIVA  
  
O fator de balanceamento do nó é dado pela diferença de altura de dir e de esq. Correto.  
O fator de balanceamento do nó e dado pela subtração do número de nós em dir com o número de nós em esq. Incorreto, o fator de balanceamento é dado pela subtração das alturas de dir e esq.  
A árvore será dita balanceada se dir tiver a mesma quantidade de nós que esq. Incorreto, a árvore será dita balanceada se o fator de balanceamento for –1, 0 ou 1.  
O fator de balanceamento do nó é dado pela diferença de altura do nó em relação à soma das alturas das árvores esq e dir. Incorreto, a definição dada não corresponde ao fator de balanceamento, que seria a diferença de altura entre dir e esq.  
As árvores esq e dir podem ter fator de balanceamento diferentes, desde que tenham a mesma altura. Incorreto, essa restrição não faz parte das definições de árvore AVL.

1. **(1 ponto)**Seja h(esq) a altura da subárvore esquerda de um nó e h(dir) a altura da subárvore direita, em uma árvore AVL, qual das alternativas a seguir é verdadeira?
   1. h(esq) - h(dir) = -1
   2. |h(esq) - h(dir)| = 1
   3. |h(esq) - h(dir)|/2 = ½
   4. |h(dir) - h(esq)| >= 1
   5. **|h(esq) - h(dir)| <= 1**

A resposta correta é: “|h(esq) - h(dir)| <=1 ”  
  
JUSTIFICATIVA  
  
A propriedade da árvore AVL é que o fator de balanceamento tem que ser +1, -1 ou 0. Nesse caso, a alternativa D é a única que captura essa propriedade: D) |h(esq) - h(dir)| <=1 .

1. **(1 ponto)**Seja a seguinte árvore:

O fator de balanceamento dos nós 15, 19, 26, 70 e 24 são:

* 1. +1, +1, 0, -1, 0
  2. +1, +1, +1, -1, 1
  3. **+1, +1, +1, 0, 0**
  4. -1, -1, -1, +1, 0
  5. -1, -1, 0, +1, 0

A resposta correta é: “+1, +1, +1, 0, 0”  
  
JUSTIFICATIVA  
  
Para computar o fator de balanceamento, precisamos computar a diferença de altura entre as subárvores direita e esquerda. Nesse caso, fazendo a computação temos:  
15 → h(dir) - h(esq) = 0 - (-1) = +1  
19 → h(dir) - h(esq) = 0 - (-1) = +1  
26 → h(dir) - h(esq) = 2 - 1 = +1  
70 → h(dir) - h(esq) = 2 - 2 = 0  
24 → h(dir) - h(esq) = 4 - 4 = 0

1. **(1 ponto)**Em uma árvore AVL, uma inserção gerou a seguinte situação parcial:  
   1. **Uma rotação para a direita no nó 25, seguida de uma rotação para a esquerda no nó 10.**
   2. Uma rotação simples para a esquerda no nó 10.
   3. Uma rotação para esquerda no nó 10, seguida de uma rotação para a direita no 25.
   4. Uma rotação simples para a direita no nó 25.
   5. Nada precisa ser feito, pois a árvore já é AVL.

A resposta correta é: “Uma rotação para a direita no nó 25, seguida de uma rotação para a esquerda no nó 10. ”  
  
JUSTIFICATIVA  
  
A árvore possui um desbalanceamento, pois o nó 25 tem fator –1 e o nó 10 tem fator +2. Como os sinais do desbalanceamento são diferentes, sabemos que precisamos de uma rotação dupla para fazer o balanceamento. A rotação dupla será composta dos passos:  
  
1 - Uma rotação para a direita no nó 25, o que fará com que o nó 15 se torne o filho à direita do nó 10 e que o 25 seja o filho à direita do nó 15.  
  
2 - Uma rotação para a esquerda no nó 10, o que fará com que o nó 10 vire o filho à esquerda do nó 15, e o 25 vire o filho à direita.

1. **(1 ponto)**Em uma árvore AVL, os nós 5, 10, 15, 20, 30 e 35 foram inseridos nessa ordem, qual a leitura pré-ordem e pós ordem da árvore resultante?
   1. 20 10 5 15 30 35 ; 5 15 10 35 20 30
   2. **20 10 5 15 30 35 ; 5 15 10 35 30 20.**
   3. 10 5 20 15 30 35 ; 5 15 35 30 20 10
   4. 5 15 10 35 20 30 ; 20 10 5 15 30 35
   5. 5 10 15 20 25 30 ; 5 15 10 35 20 30

A resposta correta é: “20 10 5 15 30 35 ; 5 15 10 35 30 20"  
  
JUSTIFICATIVA  
  
A série de inserção gera a seguinte árvore:  
  
  
Nesse caso, a leitura em pré-ordem e pós-ordem geraria.  
  
20 10 5 15 30 35 ; 5 15 10 35 30 20  
  
Lembrando que a leitura em pré-ordem é aquela em que o nó é visitado antes de se visitar as subárvores. Nesse caso, o nó raiz 20 aparece na saída, antes do nó 10. A leitura em pós-ordem fará o contrário, as subárvores esquerda e direita (nessa ordem) serão visitadas antes de visitar o próprio nó.

1. **(1 ponto)**Dada a seguinte árvore de busca AVL.

É removido o elemento 32 da árvore AVL. Assumindo que a estratégia de remoção de nós com dois filhos substitui o nó pelo sucessor lógico, qual a leitura em pré-ordem e em pós-ordem da árvore resultante?

* 1. Pré: 32 27 22 19 24 29 31 41 37 47 43  
     Pós: 19 24 22 31 29 27 37 43 47 41 32
  2. **Pré: 37 25 22 19 24 29 27 31 43 41 47  
     Pós: 19 24 22 27 31 29 25 41 47 43 37**
  3. Pré: 27 22 19 24 25 37 29 31 43 41 47  
     Pós: 19 25 24 22 31 29 41 47 43 37 27
  4. Pré: 29 22 19 24 25 37 31 32 43 41 47  
     Pós: 19 25 24 22 32 31 41 47 43 37 29
  5. Pré: 29 22 19 25 24 27 41 32 31 43 47  
     Pós: 19 24 27 25 22 31 32 47 43 41 29

A resposta correta é: “Pré: 37 25 22 19 24 29 27 31 43 41 47  
                                    Pós: 19 24 22 27 31 29 25 41 47 43 37"  
  
JUSTIFICATIVA  
  
A árvore AVL gerada após a remoção do nó 32 é:  
  
Sendo assim as leituras em pré-ordem e em pós-ordem são:  
Pré: 37 25 22 19 24 29 27 31 43 41 47  
Pós: 19 24 22 27 31 29 25 41 47 43 37  
Lembrando que a leitura em pré-ordem é aquela em que o nó é visitado antes de se visitar as subárvores. Nesse caso, o nó raiz 20 aparece na saída antes do nó 10. A leitura em pós-ordem fará o contrário, as subárvores esquerda e direita (nessa ordem) serão visitadas antes de visitar o próprio nó.

1. **(1 ponto)**Seja a árvore de busca AVL que possui as seguintes leituras em pré-ordem e em pós -ordem:

Pré: 29 22 19 25 24 27 37 32 31 43 41 47

Pós: 19 24 27 25 22 31 32 41 47 43 37 29

Qual o fator de balanceamento do nó 29 (nó raiz), do nó 22 e do nó 37?

* 1. 0    0   +1
  2. **0 +1 0**
  3. 0    0   -1
  4. 0   -1    0
  5. 0 -1 -1

A resposta correta é: “0 +1 0"  
  
JUSTIFICATIVA  
  
A árvore que possui as leituras em pré-ordem e pós-ordem dadas é da seguinte forma:  
  
  
Dada a árvore, devemos computar o fator de balanceamento dos nós:  
29 → h(dir) - h(esq) = 3 - 3 = 0  
22 → h(dir) - h(esq) = 2 - 1 = 1  
37 → h(dir) - h(esq) = 2 - 2 = 0

1. **(1 ponto)**Dado o grafo a seguir, assinale a alternativa verdadeira.
   1. O grau de B é zero.
   2. **Dois vértices possuem grau 4.**
   3. O grafo é inválido, pois self-loops não são permitidos em grafos direcionados.
   4. O grau de B é maior que o grau de D.
   5. Quatro vértices possuem o mesmo grau de saída.

A resposta correta é: “Dois vértices possuem grau 4."  
  
JUSTIFICATIVA  
  
O grau de B é zero. Incorreto, o grau de B é 4.  
O grafo é inválido, pois self-loops não são permitidos em grafos direcionados. Incorreto, self-loops são permitidos em grafos direcionados, eles não são permitidos em grafos não direcionados.  
Dois vértices possuem grau 4. Correto, os vértices B e D possuem grau 4.  
O grau de B é maior que o grau de D. Incorreto, ambos possuem grau 4.  
Quatro vértices possuem o mesmo grau de saída.Incorreto, não é possível encontrar quatro vértices com o mesmo grau de saída.

1. **(1 ponto)**Sobre formas de representação de grafos, assinale a alternativa correta.
   1. Se a nossa aplicação utilizar muitas buscas (largura ou profundidade), então é melhor utilizar matrizes de adjacências por questões de eficiência.
   2. **Se o grafo for denso, é preferível usar uma matriz de adjacências se quisermos economizar em espaço.**
   3. Se o grafo for esparso, é preferível usar uma matriz de adjacências se quisermos economizar em espaço.
   4. Se a nossa rotina mais usada precisar encontrar os predecessores de um nó, então é preferível listas de adjacências por questões de eficiência.
   5. A escolha de matrizes de adjacências ou listas de adjacências é uma questão de preferência pessoal, não interferindo no desempenho ou consumo de memória dos programas.

A resposta correta é: “Se o grafo for denso, é preferível usar uma matriz de adjacências se quisermos economizar em espaço."  
  
JUSTIFICATIVA  
  
Se o grafo for denso, é preferível usar uma matriz de adjacências se quisermos economizar em espaço. Correto  
Se a nossa aplicação utilizar muitas buscas (largura ou profundidade), então é melhor utilizar matrizes de adjacências por questões de eficiência. Incorreto, as listas de adjacências já terão os adjacentes de um nó prontos para serem usados, o que aumentaria a eficiência.  
Se o grafo for esparso, é preferível usar uma matriz de adjacências se quisermos economizar em espaço. Incorreto, seria preferível usar matrizes de adjacências se o grafo for esparso, pois as listas encadeadas para cada um dos vértices teriam poucos elementos. Se usarmos matrizes de adjacências, temos que armazenar os adjacentes em uma matriz que terá muitos elementos zeros.  
Se a nossa rotina mais usada precisar encontrar os predecessores de um nó, então é preferível listas de adjacências por questões de eficiência. Incorreto, isso demandaria uma busca por toda a estrutura. Se usarmos uma matriz de adjacências, basta olhar a coluna correspondente.  
A escolha de matrizes de adjacências ou listas de adjacências é uma questão de preferência pessoal, não interferindo no desempenho ou consumo de memória dos programas.Incorreto, a escolha interfere, e muito, no desempenho dos algoritmos e na quantidade de memória utilizada.

1. **(1 ponto)**Seja um grafo representado pela matriz de adjacências a seguir:
   1. O grafo é direcionado, ponderado, com exatamente três vértices de grau 3.
   2. **O grafo é direcionado, ponderado, com quatro vértices tendo grau maior ou igual a 5.**
   3. O grafo é não direcionado, ponderado, com exatamente três vértices de grau 3.
   4. O grafo é direcionado, não ponderado, com exatamente um vértice de grau 2.
   5. O grafo é não direcionado, ponderado, com o vértice D tendo grau 2.

A resposta correta é: “O grafo é direcionado, ponderado, com quatro vértices tendo grau maior ou igual a 5."  
  
JUSTIFICATIVA  
  
A matriz de adjacências possui valores maiores do que um, o que indica que estamos usando a matriz para armazenar pesos nas arestas, o que nos leva a concluir que o grafo é ponderado.  
O grafo é direcionado, dado que a matriz não é simétrica.  
Os vértices A, B e E possuem grau 5  
O vértice C possui grau 6.  
O vértice D possui grau 3.  
Nesse caso, a única alternativa que lista todas as características de forma correta é a letra D, o grafo é direcionado, ponderado, com quatro vértices tendo grau maior ou igual a 5.