UNIVESP
Universidade Virtual do Estado de São Paulo

GABARITO

DISCIPLINA	APLICAÇÃO	CÓDIGO DA PROVA
COM160 - Estruturas de Dados	29/07/2021	P003

QUESTÕES OBJETIVAS

Questão 1.1

Sobre a linguagem C++, assinale a alternativa correta.

a) A linguagem permite a utilização de ponteiros para uma região de memória, que pode ser alocada dinamicamente ou estaticamente. Se a região de memória for alocada dinamicamente usando o operador "new", é preciso que seja desalocada usando o "delete".

b) A linguagem é interpretada, o que faz com que seja um pouco mais lenta do que linguagens compiladas. Entretanto, mesmo sendo interpretada, é uma opção eficiente para programadores que querem escrever código rapidamente.

c) Em C++, para declarar que uma variável é do tipo ponteiro, usamos o operador "&". Nesse caso, para declarar uma variável do tipo ponteiro para inteiro e alocá-la dinamicamente fazemos: "int& variavel = new int;"

d) Em C++, quando passamos um ponteiro para uma função, esta pode ter acesso à região de memória onde os valores estão armazenados. Entretanto, por segurança, os valores não podem ser modificados.

e) C++ e Python são linguagens que possuem muitas similaridades. Por exemplo, ambas são linguagens imperativas, interpretadas, e que permitem orientação a objetos. A diferença entre ambas é que C++ não permite acesso direto à memória.

RESOLUÇÃO

A resposta correta é: A linguagem permite a utilização de ponteiros para uma região de memória, que pode ser alocada dinamicamente ou estaticamente. Se a região de memória for alocada dinamicamente usando o operador "new", é preciso que seja desalocada usando o "delete".

Justificativa

A linguagem permite a utilização de ponteiros para uma região de memória, que pode ser alocada dinamicamente ou estaticamente. Se a região de memória for alocada dinamicamente usando o operador "new", é preciso que seja desalocada usando o "delete". Correto.

A linguagem é interpretada, o que faz com que seja um pouco mais lenta do que linguagens compiladas. Entretanto, mesmo sendo interpretada, é uma opção eficiente para programadores que querem escrever código rapidamente. Incorreto, C++ é uma linguagem compilada. Na verdade, pouco do que a alternativa menciona vale realmente para C++, sendo verdade para linguagens interpretadas como a linguagem Python.

Em C++, para declarar que uma variável é do tipo ponteiro, usamos o operador "&". Nesse caso, para declarar uma variável do tipo ponteiro para inteiro e alocá-la dinamicamente fazemos: "int& variavel = new int;" Incorreto, o operador usado para declarar um ponteiro é o operador "*".

Em C++, quando passamos um ponteiro para uma função, esta pode ter acesso à região de memória onde os valores estão armazenados. Entretanto, por segurança, os valores não podem ser modificados. Incorreto, a função poderá alterar o endereço de memória, dado que possui um ponteiro para esse valor recebido por parâmetro.

C++ e Python são linguagens que possuem muitas similaridades. Por exemplo, ambas são linguagens imperativas, interpretadas, e que permitem orientação a objetos. A diferença entre ambas é que C++ não permite acesso direto à memória. C++ é uma linguagem compilada. Além disso, C++ permite sim acesso direto à memória.

Questão 1.2
Em um código, temos o seguinte trecho:

```
int *intPointer;
intPointer = new int;
```

Mais em seguida, no mesmo código, temos o comando:

```
anotherVar = *intPointer;
```

Assumindo que o código, como um todo, está compilando e executando sem problemas, assinale a alternativa correta:

a) anotherVar é uma variável do tipo ponteiro para inteiro que apontará para a mesma região de memória que intPointer.
b) anotherVar é uma variável do tipo int. Se mudarmos posteriormente o conteúdo de anotherVar, também mudaremos o conteúdo da região de memória apontada por intPointer.
c) anotherVar é do tipo ponteiro para inteiro. Entretanto, anotherVar aponta para uma região de memória diferente daquela apontada por intPointer, dado que a última instrução fez apenas a cópia de uma região de memória para outra.
d) anotherVar é uma variável do tipo int. Se mudarmos posteriormente o conteúdo de anotherVar, não mudaremos o conteúdo de intPointer, dado que a última instrução fez apenas a cópia de uma região de memória para outra.
e) anotherVar é uma variável do tipo ponteiro para inteiro alocada de forma dinâmica. Nesse caso, não apontará para a mesma região de memória de *intPointer.

RESOLUÇÃO
A resposta correta é: anotherVar é uma variável do tipo int. Se mudarmos posteriormente o conteúdo de anotherVar, não mudaremos o conteúdo de intPointer, dado que a última instrução fez apenas a cópia de uma região de memória para outra.

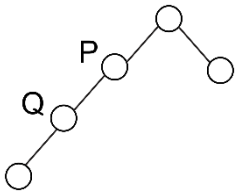
Justificativa
Como anotherVar recebeu o conteúdo de uma região de memória que deveria conter um valor inteiro, então anotherVar só pode ser do tipo inteiro (dado que o programa compila normalmente como informado no enunciado da questão). Nesse caso, anotherVar recebe apenas uma cópia do conteúdo

que estava na posição de memória apontada por intPointer, não havendo um compartilhamento de região de memória entre anotherVar e *intPointer. Em outras palavras, temos no trecho de código duas regiões de memória independentes que possuem o mesmo conteúdo.

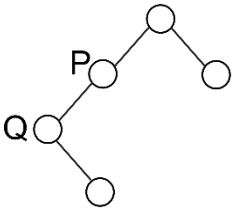
Questão 1.3

Em se tratando dos conceitos de árvore AVL, em qual das alternativas a seguir temos um nó P desbalanceado, pai de um nó Q, tal que o balanceamento para árvore AVL necessita de uma rotação de Q para a esquerda seguida de uma rotação de P para a direita.

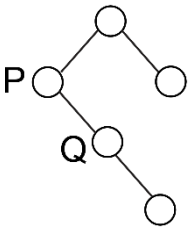
a)



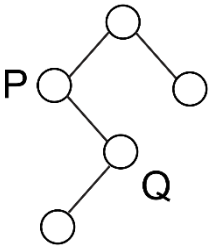
b)



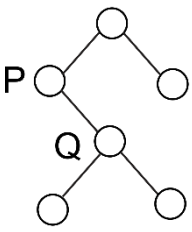
c)



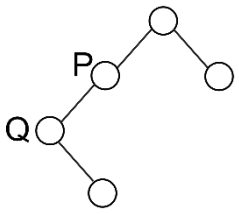
d)



e)



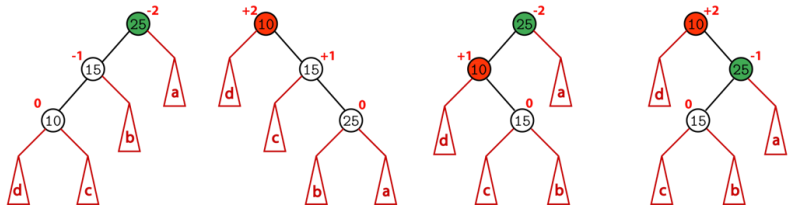
RESOLUÇÃO

A resposta correta é: 

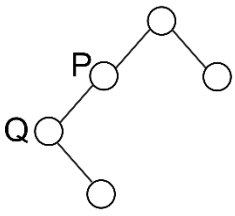
Justificativa

Para termos uma rotação de Q para a esquerda seguida de uma rotação de P para a direita, precisamos que o fator de balanceamento de Q seja igual a +1 e que o fator de balanceamento de P seja igual a –2. Lembrando que o fator de balanceamento de um nó é dado pela subtração da altura da subárvore da direita pela altura da subárvore da esquerda. A tabela a seguir mostra todas as rotações em função do fator de balanceamento e apresenta exemplos com as rotações computadas:

Nó Desbalanceado	Filho do Nó Desbalanceado	Tipo de Rotação
+2	+1	Simples à Esquerda
+2	0	Simples à Esquerda
+2	-1	Dupla com Filho para a Direita e pai para Esquerda
-2	+1	Dupla com Filho para a Esquerda e pai para Direita
-2	0	Simples à Direita
-2	-1	Simples à Direita

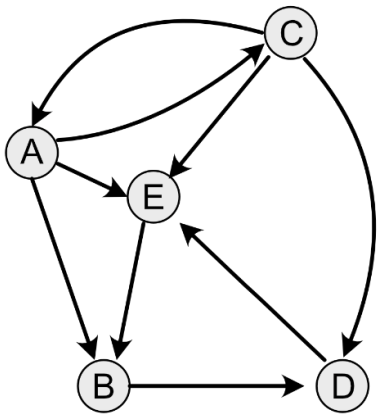


Dessa forma, a situação requerida é encontrada apenas na configuração a seguir:



Questão 1.4

Sobre o grafo apresentado a seguir, assinale a alternativa verdadeira.



a. Uma possível busca em largura, iniciando no vértice A, poderia visitar os vértices na seguinte ordem: A, B, C, D, E.
b. Uma possível Busca em profundidade, iniciando no vértice C, poderia visitar os vértices na seguinte ordem: C, D, E, B, A.
c. Uma busca em profundidade neste grafo utilizando o algoritmo visto em aula, iria utilizar uma fila para gerenciar os backtrackings.
d. Uma possível busca em largura, iniciando no vértice C, poderia visitar os vértices na seguinte ordem: C, E, B, D, A.
e. Uma possível busca em profundidade, iniciando no vértice A, poderia visitar os vértices na seguinte ordem: A, C, D, B, E.

RESOLUÇÃO
A resposta correta é: Uma possível Busca em profundidade, iniciando no vértice C, poderia visitar os vértices na seguinte ordem: C, D, E, B, A.

Justificativa
Uma possível busca em largura, iniciando no vértice A, poderia visitar os vértices na seguinte ordem: A, B, C, D, E. Incorreto, em uma busca e largura devemos visitar os nós camada por camada. Nesse caso, ao iniciar no nó A, devemos visitar primeiramente os nós B, C e E (em qualquer ordem). Entretanto, na alternativa, o nó D está sendo visitado antes do nó E, o que indica que a visitação não pode ter ocorrido camada por camada.

Uma possível Busca em profundidade, iniciando no vértice C, poderia visitar os vértices na seguinte ordem: C, D, E, B, A. Correto, na busca em profundidade, a estratégia consiste em se aprofundar no grafo sempre que possível. Nesta alternativa, começamos no nó C e seguimos para um dos seus filhos, que seria o nó D. O único filho de D é o E, sendo a única opção. De E, temos que seguir para B, única opção. Como não temos mais nenhum filho de B, devemos fazer backtrackings até alcançar o nó C (origem da busca), para que o segundo filho de C seja visitado, o nó A.

Uma busca em profundidade neste grafo utilizando o algoritmo visto em aula, iria utilizar uma fila para gerenciar os backtrackings. Incorreto, o algoritmo de busca em profundidade precisa utilizar uma pilha para gerenciar os backtrackings. Nesse caso, sempre que visitamos um nó, adicionamos os filhos na pilha para, posteriormente, desempilhar os filhos de acordo com a ordem de visitação da busca em profundidade.

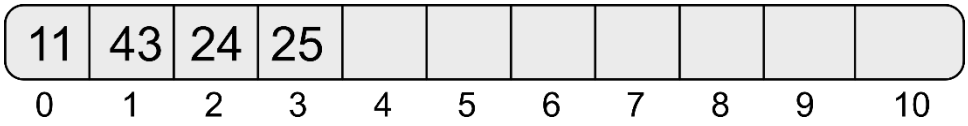
Uma possível busca em largura, iniciando no vértice C, poderia visitar os vértices na seguinte ordem: C, E, B, D, A. Incorreto, o algoritmo de busca em largura precisa visitar os nós camada por camada. Assim, após visitar o nó C, a próxima camada é composta por {A, D, E}, o que nos leva a concluir que o nó B não poderia ter sido visitado antes da visitação desses três elementos, como apresentado na alternativa.

Uma possível busca em profundidade, iniciando no vértice A, poderia visitar os vértices na seguinte ordem: A, C, D, B, E. Incorreto, o algoritmo de busca em largura precisa visitar os nós camada por camada. Assim, após visitar o nó A, a próxima camada é composta por {B, C, E}, o que nos leva a concluir que o nó D não poderia ter sido visitado antes da visitação desses três elementos, como apresentado na alternativa.

QUESTÕES DISSERTATIVAS

Questão 2

Em uma pilha, implementada internamente como um vetor, temos após algumas inserções e remoções a seguinte configuração do vetor.



A partir dessa configuração, fazemos as seguintes operações:
Pop()
Push(19)
Push(32)
Pop()
Push(34)

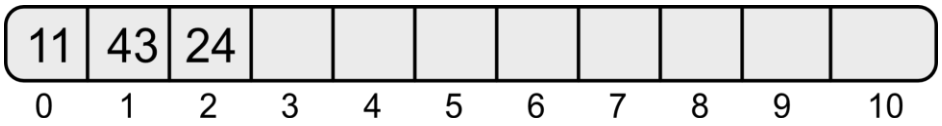
Feito isso, responda:

1. Quais os elementos retornados pelas duas operações Pop(), em ordem?
2. Quais as configurações do vetor interno da estrutura de dados após cada comando?

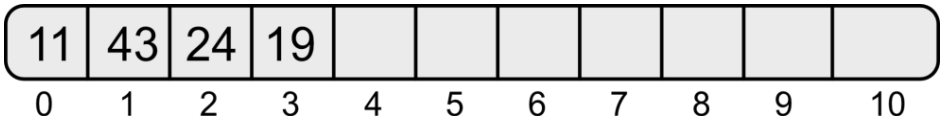
RESOLUÇÃO

1 - Os elementos retornados pelo Pop serão: **25 e 32.**
2 - Após cada operação, teremos os seguintes vetores internos:

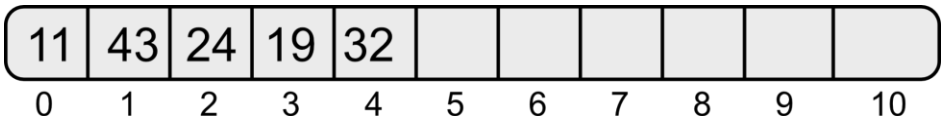
Pop()



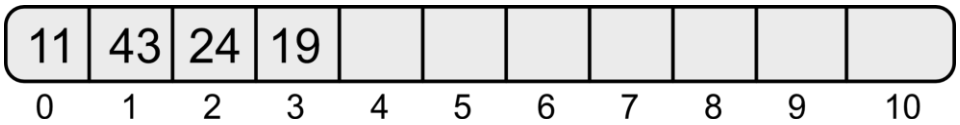
Push(19)



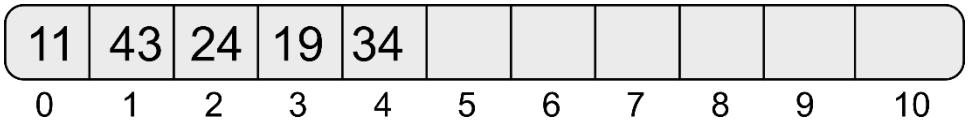
Push(32)



Pop()



Push(34)

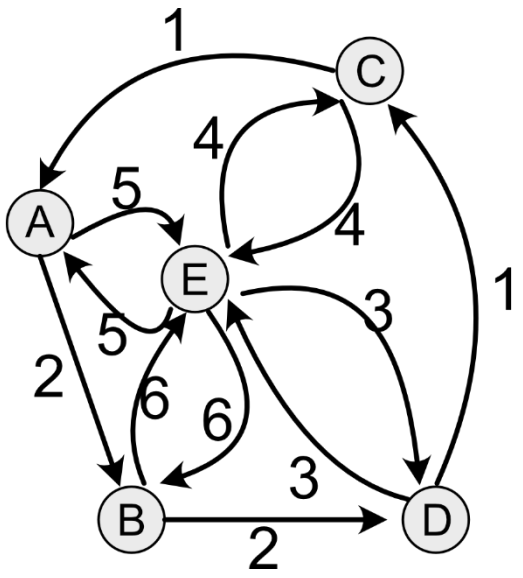


Rubricas | critérios de correção

Fornecer 40% da nota para quem acertou a primeira questão e 60% da nota para quem acertou a segunda questão. Como são 5 vetores parciais, cada um deles determinará 12% do peso da nota final. Se o aluno errar qualquer posicionamento de valores dentro de um vetor interno, descontar os 12% independente do erro cometido.

Questão 3

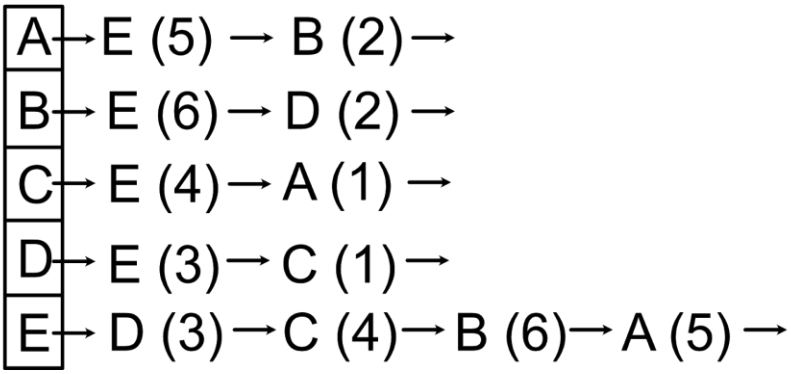
Dado o grafo a seguir, mostre a sua representação utilizando matrizes de adjacências e listas de adjacências:



RESOLUÇÃO

A resolução consiste em identificar, usando matrizes de adjacências e listas de adjacências, todos os vértices e arestas. Em ambas as representações, é preciso identificar os pesos nas arestas de alguma forma:

	A	B	C	D	E
A	0	2	0	0	5
B	0	0	0	2	6
C	1	0	0	0	4
D	0	0	1	0	3
E	5	6	4	3	0



Rubricas | critérios de correção

Pontuar 50% da questão para as matrizes de adjacências e 50% da questão para a lista de adjacências. Nas listas de adjacências, o aluno deve indicar que o peso da aresta ficará em algum nó da lista encadeada, isso pode ser feito de várias formas, sugiro considerar qualquer forma de representação em que fique claro que o peso está acoplado ao elemento da lista encadeada. Em matrizes de adjacências, os pesos ficam na própria matriz.

Descontar 10% da questão por cada erro ou omissão inserido pelo aluno. Nesse caso, considerar erro com o seguinte critério:

Matriz de Adjacência: uma célula contendo um valor diferente do mostrado acima caracteriza um erro.
Coleção de Listas de Adjacências: um elemento adicionado inadvertidamente em uma lista deve ser considerado um erro. Além disso, considerar erro um elemento que deveria estar na lista, mas não está.