Universidade do Minho	**Escola de Engenharia da Universidade do Minho** Mestrado Integrado em Eng. Electrónica Industrial e Computadores Complementos de Programação de Computadores	**2012/2013** **MIEEIC** **(1º Ano)** **2º Sem**
Teste Nº2 - Época Normal - Data 07/06/2013, Duração 1h45m		

Nome:_____ Nº Aluno: _____

Responda às seguintes questões, preenchendo a tabela com a **opção correcta (em maiúsculas)** **(Correcto:x Val / Errado: -x/3 Val).** Suponha que foram realizados as inclusões das bibliotecas necessárias.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

GRUPO I (6 Valores)

1. Considere o seguinte fragmento de código C++:

```
vector<int> v1; int n; cin >> n;
for(int i=1; i<n; i*=2)
   for(int j=i; j<n; j++) {
       cout << i << " - " << j << endl;
   }
```

A complexidade temporal do fragmento de código é:

A) T(n)=O(n^2).
B) T(n)= O(2*n).
C) T(n)=O(n*log(n)).
D) T(n)=O(n).
E) Nenhuma das anteriores.

2. Considere o seguinte fragmento de código C++:

```
vector<int> v1; int n; cin >> n;
for(int i=1; i<n; i++) {
   for(int j=0; j<n; j*=2)
       cout << i << " - " << j << endl;
   }
```

A complexidade temporal do fragmento de código é:

A) T(n)=O(n^2).
B) T(n)= O(1)
C) T(n)=O(n*log(n)).
D) T(n)=O(n).
E) Nenhuma das anteriores.

3. Considere o seguinte fragmento de código C++:

```
vector<int> v1; int n; cin >> n;
for(int i=0; i<n; i++)
   for(int j=1; j<2; j++) {
     for(int k=1; k<i; k++) {
       cout << i << j << k << endl;
   }
```

A complexidade Espacial do fragmento de código é:

A) S(n)=O(n^3).
B) S(n)= O(1).
C) S(n)= O(n).
D) S(n)=O(n^2).
E) Nenhuma das anteriores.

4. Considere o seguinte fragmento de código C++:

```
vector<int> v1; int n; cin >> n;
for(int i=0; i<n; i++)
   for(int j=i; j<i+2 ; j++)
      for(int k=i+2; k>2 ; k = k/2)
         v1.push_back(i+j+k);
for(vector<int>::iterator it = v1.begin();
    it!=v1.end(); it++)
        cout << *it << " ";
```

A complexidade temporal do fragmento de código é:

A) A complexidade temporal é T(n)=O(n^3).
B) A complexidade temporal é T(n)=O(n*log(n)).
C) A complexidade temporal é T(n)=O(n^2*log(n)).
D) A complexidade temporal é T(n)=O(n^2).
E) Nenhuma das anteriores.

5. Considere o seguinte fragmento de código C++:

```
vector<int> v1; int n; cin >> n;
for(int i=0; i<n; i=i+3)
   for(int j=i; j<i+2; j++) {
       cout << i << " - " << j << endl;
   }
```

A complexidade temporal do fragmento de código é:

A) T(n)=O(n^2).
B) T(n)= O(2*n).
C) T(n)=O(n*log(n)).
D) T(n)=O(n).
E) Nenhuma das anteriores.

6. Considere o seguinte programa:

```
// fichs inclusão e declarações gerais
int f(vector<int> arr) {
  queue<string> q;
  stack<string> s;
  if (n < 1) return -1;
  for (int i=0; i < arr.size(); i++) {
     q.push(arr[i]);
  }
  for ( ; !q.empty(); ) {
     s.push(q.front()); q.pop();
  }
  for ( ; !s.empty(); ) {
     cout << s.top() << " "; s.pop();
  }
  return 0;
}
```

A) No final do programa a fila estará vazia mas a pilha não (pois recebeu todos os elementos da fila).
B) Se função for invocada com um arr={}, isto é um vetor vazio, o programa "estoura".
C) Se a função for invocada com um arr={2, 3, 4}, aparecerá no ecrã: 2 3 4
D) No final da execução, tanto a pilha como a fila terão size 1 (tamanho 1).
E) Nenhuma das respostas anteriores.

7. A classe *queue* da STL de C++ é um tipo abstrato de dados.

A) Não, pois é um dos tipos base de dados do C++ (tal como a pilha - stack).
B) Sim, mas só se se declarar no código fonte que queue é do tipo abstrato.
C) Não, pois o tipo de dados abstrato é a lista (list da STL).
D) Sim, pois contém dados e operações sobre esses dados.
E) Não pois encapsula os dados e operações.

8. Supondo o seguinte código aplicado a uma árvore binária, selecione a afirmação verdadeira:

```
1 void trav(TNode *X) {
2    if(X!=0) {
3        trav(X->right());
4        trav(X->left());
5        //usar nodo
6    }
7 }
```

A) Se a linha 5 for substituída por `delete X;` poderia ser utilizado para apagar completamente a árvore binária.
B) O código implementa uma travessia em pós-ordem de uma árvore binária.
C) O código implementa uma travessia em pré-ordem de uma árvore binária.
D) O código implementa uma travessia em ordem de uma árvore binária.
E) Nenhuma das anteriores.

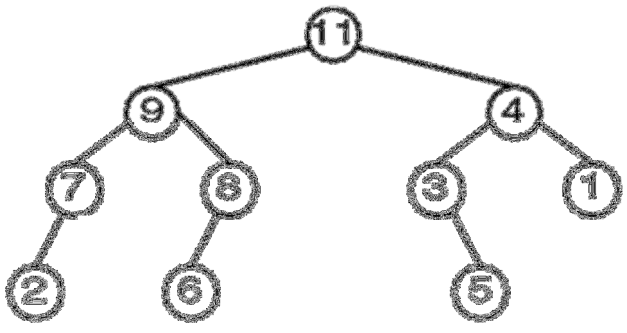
9. A operação de inserção de um elemento numa pilha

A) Não possui complexidade temporal pois é imediata;
B) Possui complexidade temporal constante quer a implementação da pilha seja baseada em vectores quer seja baseada em listas ligadas;
C) Possui complexidade temporal linear se a implementação da pilha é baseada em vectores e constante se baseada em listas ligadas;
D) Possui complexidade temporal linear quer a implementação da pilha seja baseada em vectores ou em listas ligadas;
E) Nenhuma das anteriores.

10. A operação de remoção de um elemento numa lista:

A) Não é possível em vetores pois não podem ficar "buracos" no vetor.
B) Possui complexidade temporal linear se a implementação da lista é baseada em vetores, e constante se baseada em listas ligadas.
C) Possui complexidade temporal linear se a implementação da lista é baseada em vetores ou em listas ligadas.
D) Possui complexidade temporal constante se a implementação da lista é baseada em vetores ou em listas ligadas.
E) Nenhuma das anteriores.

Suponha a seguinte árvore binária:



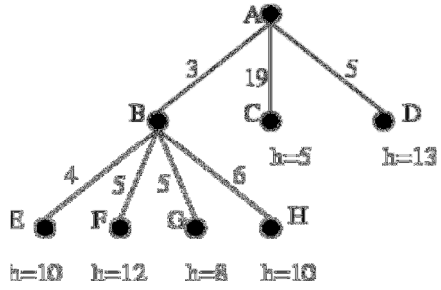
11. A travessia em ordem da árvore resulta na seguinte sequência:

A) 2 7 9 6 8 11 4 3 5 1
B) 11 9 7 2 8 6 4 3 5 1
C) 2 7 9 6 8 11 3 5 4 1
D) 2 7 9 8 6 11 3 5 4 1
E) Nenhuma das anteriores.

12. A travessia pós-ordem da árvore resulta na seguinte sequência:

A) 2 7 8 6 9 3 5 1 4 11
B) 2 7 9 6 8 11 3 5 4 1
C) 11 9 7 2 8 6 4 3 5 1
D) 11 9 7 2 6 8 4 3 5 1
E) Nenhuma das anteriores.

13. Supondo a seguinte árvore de pesquisa em que cada arco apresenta o custo do operador correspondente e h é uma função heurística que estima um custo para a solução, diga qual o nó expandido em seguida utilizando cada um dos seguintes métodos: i) Pesquisa em largura; ii) Pesquisa em Profundidade;



A) i) Nó C ii) Nó D
B) i) Nó C ii) Nó C
C) i) Nó E ii) Nó C
D) i) Nó C ii) Nó E
E) Nenhuma das anteriores.

14. Diga qual o nó expandido em seguida utilizando a Pesquisa Gulosa/Gananciosa;

A) Nó G
B) Nó C
C) Nó D
D) Nó E
E) Nenhuma das anteriores.

15. Diga qual o nó expandido em seguida utilizando o Algoritmo A*:

A) Nó G
B) Nó C
C) Nó D
D) Nó E
E) Nenhuma das anteriores.

16. O método de ordenação por partição (Quick Sort):

A) No caso médio tem uma complexidade no espaço de ordem inferior à dos métodos de ordenação por inserção e ao Bubble Sort pois faz a partição e ocupa menos espaço.
B) No caso médio (valores aleatoriamente distribuídos) tem complexidade no tempo O(n*log(n)) e complexidade no espaço O(n*log(n)).
C) É sempre mais rápido do que a ordenação por inserção e quase sempre mais rápido do que a ordenação MergeSort.
D) Escolhendo para Pivot sempre o maior valor do array, tem complexidade no tempo $O(n^2)$ e complexidade no espaço O(n).
E) Nenhuma das anteriores.

GRUPO II

2) Pretende-se implementar um programa para gestão de uma biblioteca. Considere as classes **CBiblioteca** e **CLivro** a seguir definidas (estas classes estão incompletas).

```
class CBiblioteca {
       vector <Leitor*> leitores;
       vector <CLivro> livros;
   public:
       CBiblioteca();
       void adicionaLeitor(Leitor &l1);
       void adicionaLivro(CLivro &l1);
       CLivro &getLivro(string &tituloL);
};
class CLivro {
       string titulo, autor;
       string ISBN;
       int numExemplares;
       CLivro &operator+ (const CLivro &livro1);
};
```

2.1) Construa a classe Leitor, que contém os atributos privados *nome* (string) e *codigo* (inteiro). Os leitores devem ter um código único e sequencial.

2.2) Implemente o respectivo método construtor e o método abstracto *double getMensalidade.*

2.3) Na classe *CBiblioteca*, implemente o método *CLivro& getLivro(string &tituloL)*, que retorna o livro cujo título é *tituloL.*

2.4) Na classe *CLivro*, implemente o operador +. A soma de dois livros só pode ser realizada se os livros forem iguais (tiverem o mesmo ISBN). Nesse caso, a soma é um outro livro com o titulo, autor e ISBN do primeiro, e número de exemplares igual à soma do número de exemplares dos dois livros.

2.5) Suponha que os leitores podem ser de dois tipos: sócios e não sócios. Construa a subclasse *LeitorSocio***.** Os leitores sócios pagam uma determinada mensalidade à biblioteca, e podem requisitar um número ilimitado de livros. O valor da mensalidade é um argumento do construtor. Implemente o método construtor e o método *getMensalidade()*.

2.6) Construa a subclasse *LeitorNaoSocio***.** Os leitores não sócios possuem um número limite de livros que podem requisitar (argumento do construtor). Esta classe possui também o atributo *livrosReq* que indica o número de livros que requisitou nesse mês, e o atributo *taxa* (taxa de empréstimo que é um argumento do construtor). A mensalidade a pagar é igual a *livrosReq*taxa*. Implemente o método construtor e o método *getMensalidade()*.

2.7) Na classe *CBiblioteca*, implemente o método *getMensalidade()*, que retorna o valor da mensalidade paga por todos os leitores da biblioteca.

2.8) Considere que pretendia realizar a operação de requisição de um livro por parte de um leitor Especifique quais as estruturas de dados que usaria, alterando/adicionando os atributos das classes que entender. Explique como realizaria a operação de requisição de um livro. Não precisa apresentar código.

2.9) Suponha que deseja copiar a informação completa relativa a todos os livros da biblioteca para uma fila de modo a serem posteriormente processados. Apresente a declaração de tal fila e o código que lhe permite efectuar a referida cópia.

GRUPO III

3.1) Implemente os seguintes membros-função para a classe `Queue` descrita nos slides/aulas da disciplina.

a) `entra_e_sai(int n)`: retira os primeiros n elemento da fila e coloca-os no final da fila pela mesma ordem.

c) `eliminarN(int n)`: elimina os n primeiros elementos da fila

3.2) Implemente o membro-função `concatena` para a classe *`stack`*, descrita nos slides/aulas da disciplina que concatena duas pilhas numa única esvaziando a outra. Os elementos são copiados um por um de uma das pilhas para a outra e retirados da pilha inicial.

3.3) Suponha a classe `List` descrita nos slides/aulas da disciplina. Implemente um método para intercalar numa lista os elementos de outra. A segunda lista não é modificada.

```
void List::intercalar(const List &lst)
```

Exemplo: [a b c d e f g] intercalada com [v w x y z] passaria a conter [a v b w c x d y e z f g].

3.4) Na classe BTree descrita nos slides/aulas da disciplina implemente o método: `int escreverN(const int n);`

Este método percorre uma árvore binária contendo números inteiros, em ordem, e envia para o `ecrã`, os primeiros n valores que encontrar na árvore. Deve também escrever no ecrã e retornar a soma desses valores.