Universidade do Minho	**Escola de Engenharia da Universidade do Minho** Mestrado Integrado em Eng. Electrónica Industrial e Computadores Complementos de Programação de Computadores	**2015/2016** **MIEEIC** **(1º Ano)** **2º Sem**
Docentes: Luís Paulo Reis, Isabel Moura		
Exame Nº2 - Época Normal - Data 03/06/2016, Duração 1h45m+10min (com consulta)		

Nome:_____ Nº Al:_____

Responda a todas as questões utilizando exclusivamente as folhas agrafadas fornecidas. **Não escreva nada em qualquer outra folha durante a realização do exame.** Suponha que foram realizados as inclusões das bibliotecas necessárias (#include"). Caso considere que a informação fornecida é insuficiente indique os pressupostos que assumiu na sua folha de teste.

Responda às seguintes questões, preenchendo a tabela com a **opção correcta (em maiúsculas)** (Correcto:x Val / Errado: -x/3 Val).

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

GRUPO I (6.0 Valores)

1. Considere o seguinte fragmento de código C++. O que se obtém como resultado da execução do código, no ecrã?

```
int i, n=6;
vector<int> v1;
for(i=n; i>1; i--) v1.push_back(i);
for(vector<int>::iterator it = v1.begin();
     it!=v1.end(); it++)
          cout << *it << " " ;
```

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

2. Considere a estrutura de dados "Pilha". Qual das seguintes afirmações é incorreta?

A. A pilha é uma estrutura de dados FIFO (first-in-first-out).
B. Novos elementos apenas podem ser adicionados ao topo da pilha.
C. A operação de remoção de um elemento da pilha é efetuada em tempo constante.
D. A pilha pode ser implementada recorrendo ao uso de listas ligadas.
E. Nenhuma das possibilidades anteriores.

3. Considere um vetor ordenado de forma não decrescente. Um algoritmo percorre o vetor, do início ao fim e insere os elementos lidos do vetor numa pilha. Ao retirar os elementos da pilha, obtém-se:

A. Uma sequência de elementos por ordem não decrescente
B. Uma sequência de elementos não repetidos por ordem decrescente
C. Uma sequência de elementos não repetidos por ordem crescente
D. Uma sequência de elementos por ordem não crescente
E. Nenhuma das possibilidades anteriores

4. Num restaurante de Hamburgers, os diferentes tipos de hamburgers são dispostos, pelo cozinheiro, em diferentes colunas de um tabuleiro inclinado, de modo a que os empregados consigam aceder rapidamente aos pedidos dos seus clientes. O cozinheiro coloca os hambúrgueres no topo da respectiva coluna e o empregado retira o hambúrguer na base da coluna. Qual a estrutura de dados mais adequada a usar na gestão de atendimento de clientes?

A. `vector<stack<Hamburger>>` (isto é um vetor de pilhas de hambúrgueres)
B. `queue<queue<Hamburger>>` (isto é uma fila de filas de hambúrgueres)
C. `stack<stack<Hamburger>>` (isto é uma pilha de pilhas de hambúrgueres)
D. `vector<queue<Hamburger>>` (isto é um vetor de filas de hambúrgueres)
E. É completamente indiferente

5. Considere uma lista ordenada, circular e duplamente ligada. O primeiro elemento da lista é o elemento menor da lista ordenada. Qual a complexidade temporal da operação de encontrar o maior elemento da lista?

A. O(1)
B. $O(N^2)$
C. O(logN)
D. O(N)
E. Nenhuma das possibilidades anteriores

6. O algoritmo ordenação por seleção é usado na ordenação, por ordem crescente, de um vetor de 10 elementos. Considerando que o vetor está inicialmente ordenado por ordem decrescente {10,9,8,...,1}, quantas trocas inúteis são realizadas, aproximadamente? (uma troca inútil ocorre quando um elemento é trocado com ele próprio, logo o vetor não se altera).

A. 5
B. 0
C. 3
D. 10
E. Nenhuma das possibilidades anteriores

7. Usando um algoritmo de ordenação por inserção, um vetor de 400 000 elementos é ordenado em 10ms. A ordenação de um novo vetor de 800 000 elementos (isto é com o dobro dos elementos) demorará, aproximadamente?

A. 30 - 50 ms
B. 10 - 15 ms
C. 200 – 400 ms
D. 100 - 200 ms
E. Nenhuma das possibilidades anteriores

8. Indique a complexidade temporal do seguinte fragmento de código:

```
void funcaoEx1(vector<int> &v) {
     for(int i=0; i< v.size(); i++)
          v[i]*=2;
     for(int j=0; j< v.size()/2; j++)
          v[j]-=1;
}
```

A. O(logN)
B. O(N*logN)
C. O(N^2)
D. O(N)
E. Nenhuma das possibilidades anteriores

9. Indique a complexidade temporal do seguinte fragmento de código:

```
void funcaoEx2(vector<int> &v) {
     for(int i=0; i< v.size(); i++) {
          v[i]*=2;
          for(int j=0; j< v.size()/2; j++)
             v[j]-=1;
     }
}
```

A. O(logN)
B. O(N*logN)
C. O(N^2)
D. O(N)
E. Nenhuma das possibilidades anteriores

10. O algoritmo de pesquisa binária, normalmente é usado na pesquisa de um valor num vetor ordenado. Por engano, este algoritmo foi usado no seguinte vetor desordenado de 7 elementos: [1, 3, 2, 5, 13, 8, 21]. Qual dos seguintes valores NÃO é encontrado pelo algoritmo?

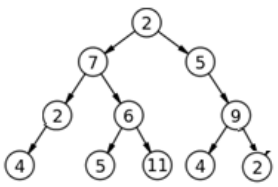
A. 3
B. 13
C. 21
D. 1
E. Nenhuma das possibilidades anteriores

11. Sobre a escolha do elemento pivot no algoritmo de ordenação quickSort, é verdade que:

A. Escolher o maior valor do vetor garante complexidade temporal O(N*logN).
B. Escolhendo a mediana de quaisquer 3 valores é garantida a complexidade temporal O(N^3).
C. Está sempre garantida a complexidade temporal O(N).
D. Escolher o menor valor do vetor garante complexidade temporar O(1).
E. Nenhuma das possibilidades anteriores

12. No método de ordenação por partição (Quick Sort) para ordenar um vetor X com 999 elementos sendo que esses elementos são garantidamente os números de 1 a 999, como se pode definir o valor para o primeiro pivot?

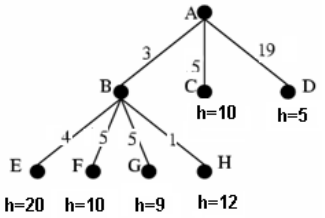
A) X[999]
B) (1+999)/2
C) 999
D) X[500]
E) Nenhuma das Anteriores



13. Suponha a seguinte árvore binária (não balanceada). Indique o que é escrito no écran executando a função trav passando-lhe com parâmetro o nó raiz (topo) da árvore.

```
void trav(TNode *N) {
 if(N!=0) {
   trav(N->right());
   trav(N->left());
   cout << N->value() <<" ";//imprime valor do nó
 }
}
```

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



14. 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 Gulosa; ii) Pesquisa em Largura

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

15. Diga quais os nós expandido em seguida utilizando respetivamente a pesquisa i) Custo Uniforme e a ii) A*.

A) i) Nó C ; ii) Nó H
B) i) Nó H ; ii) Nó C
C) i) Nó C ; ii) Nó G
D) i) Nó H ; ii) Nó H
E) Nenhuma das anteriores

GRUPO II (10.0 Valores)

Uma agência de aluguer de viaturas pretende implementar um sistema de gestão dos seus clientes e viaturas. A classe `CAgencia` guarda informação sobre os seus clientes, frota de viaturas da agência e informação das viaturas alugadas. A informação sobre um Cliente (classe `CCliente`) inclui o nome, a idade, viatura alugada e data (ano) do último aluguer, se existir. Cada cliente só pode alugar uma viatura de cada vez. A informação sobre uma Viatura (classe `CViatura`) inclui a marca, matrícula e informação sobre existência de ar condicionado. O membro-dado alugueres guarda informação sobre o número de vezes que uma dada viatura já foi alugada. As classes `CAgencia`, `CViatura` e `CCliente` estão parcialmente definidas a seguir. Suponha que os métodos indicados nas classes se encontram já implementados.

```
class CAgencia {
   vector<CViatura*> viaturas;
   vector<CCliente> clientes;
   vector<CViatura*> v_alugadas;
public:
   CAgencia();
   ~CAgencia();
   vector<CCliente> getClientes() const;
   vector<CViatura*> getViaturas() const;
   int numClientes() const;
   int numViaturas() const;
   int numViaturasAlugadas() const;
};
class CViatura {
   string marca;
   string matricula;
   bool ar_condicionado;
   int alugueres;
public:
   CViatura(string mc, string mt, bool arc);
   string getMarca() const;
   string getMatricula() const;
   int getAlugueres() const;
   void addAluguer();
};
```

```
class CCliente {
   string nome;
   unsigned int idade;
   CViatura *viatura;
   int anoUltAluguer;
public:
   unsigned int getIdade() const;
   int getAnoUltAluguer() const;
   bool setViatura(CViatura *v);
   CViatura *getViatura() const;
};
```

2.1) Defina e implemente na classe `CCliente` o construtor, o método `setNome` e o método `getNome`.

2.2) Implemente na classe `CAgencia` o membro-função:
`int existeCliente(string nomeCli) const`
O método `existeCliente` retorna o índice do cliente de nome `nomeCli` no vetor de clientes da agência. Se o cliente não existir, retorna -1.

2.3) Implemente na classe `CAgencia` o membro-função:
`bool adicionaCliente(string nomeCli, unsigned int idadeCli);` Esta função adiciona o cliente de nome `nomeCli` ao vetor clientes da agência. Se o cliente não tiver idade igual ou superior a 18 anos ou se cliente já existir, deve retornar falso sem adicionar o cliente e imprimir no écran qual o erro correspondente "Menor de Idade!" ou "Já Existente!".

2.4) Implemente na classe CAgencia os métodos seguintes:
`int existeViatura(string &mat) const`
`bool existeViaturaAlugada(Viatura *vt1) const`
O Método `existeViatura` deve retornar a posição da viatura com matricula `mat` no vetor de viaturas (ou -1 caso não exista). O método `existeViaturaAlugada`, dado um apontador para uma viatura deve retornar verdadeiro/falso dependendo se essa viatura está alugada (isto é consta do vetor de viaturas alugadas).

2.5) Implemente na classe `CAgencia` o membro-função:
`bool aluga(string nomeCli, string mt_viatura, int anoAtual);` Esta função realiza o aluguer da viatura com matrícula `mt_viatura` pelo cliente de nome `nomeCli` na data `anoAtual`. O vetor `v_alugadas` mantém as viaturas que estão atualmente alugadas (note que o vetor `viaturas` inclui todas as viaturas da agência, alugadas ou não). Esta operação implica ainda a atualização do vetor `v_alugadas` e o incremento do membro-dado `alugueres`. No caso da viatura com matrícula `mt_viatura` já estar alugada ou não existir, a função deve retornar falso.

2.6) Implemente na classe Agencia o membro-função: `void ordenaViaturas() que` ordena o vetor `viaturas` por ordem decrescente de número de alugueres de cada viatura. Viaturas com o mesmo número de alugueres são ordenadas por ordem alfabética (decrescente) de matrícula. Na ordenação use o método de ordenação por seleção estudado nas aulas.

2.7) Implemente na classe `CAgencia` o membro-função: `void clientesAntigos(int anoAl, string nomFich)` que grava no ficheiro `nomFich`, a informação de todos os clientes cujo último aluguer (membro-dado `anoUltAluguer`) seja anterior ao ano `anoAl`.

2.8) Implemente na classe `CAgencia` o membro-função:
`bool devolve(string nomeCli, string mt_viat)`
A função devolve à agência de aluguer a viatura com matrícula `mt_viat` que estava alugada pelo cliente `nomeCli`. Aquando da devolução da viatura, esta deve ser removida do cliente e do vetor de viaturas alugadas. Se o cliente `nomeCli` não alugou a viatura de matrícula `mt_viatura`, a função retorna falso.

2.9) Implemente uma função `main()` que lhe permita testar adequadamente as 8 alíneas anteriores.

GRUPO III (4.0 Valores)

3.1) Implemente a função genérica:
```
template <class T>
vector<T> unicosPilha(stack<T> &p1,stack<T> &p2)
```
Esta função determina e retorna um vetor com todos os elementos de uma pilha p1 que não existem também na pilha p2. Considere que ambas as pilhas p1 e p2 não contêm elementos repetidos.

3.2) Suponha a classe `List` descrita nos slides/aulas da disciplina. Implemente um método para adicionar os elementos de uma segunda lista à primeira lista. A segunda lista não é modificada: `void List::adiciona(const List &lst);`
Exemplo: a adição de L1=[a b c d e f g] com L2= [v w x y z] resultaria em L1= [a b c d e f g v w x y z].

3.3) Na classe `BTree` descrita nos slides/aulas da disciplina implemente o método `int eliminarFolhas();` Este método percorre a árvore e elimina todos os nós da árvore que não tenham filhos e retorna o número de nós eliminados. Por exemplo, na árvore ao lado seriam eliminados os nós: 7, 8, 9 e 10. Nota: Cuidado com a ordem de travessia da árvore.