Universidade do Minho	**Escola de Engenharia da Universidade do Minho** Mestrado Integrado em Eng. Electrónica Industrial e Computadores Complementos de Programação de Computadores	**2013/2014** **MIEEIC** **(1º Ano)** **2º Sem**
Docentes: Luís Paulo Reis, Pedro Pimenta e C. Filipe Portela		
Teste Nº2 - Época Normal - Data 04/06/2014, Duração 1h45m+10min (com consulta)		

Nome:_____ Nº Aluno:_____

Responda às seguintes questões, preenchendo a tabela com a **opção correta (em maiúsculas)** **(Correto: x Val / Errado: -x/3 Val).** Suponha que foram realizados as inclusões das bibliotecas necessárias e "using namespace std;".

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

GRUPO I (7 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<10; i++) {
   for(int j=1; 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=0; j<10; j++) {
     for(int k=1; k<j; 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=0; 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. Qual a complexidade temporal da seguinte função:

```
int factorial(int n){
     if(n==0) return 1;
     else return n*factorial(n-1);
}
```

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

6. Qual a complexidade espacial da função anterior?

A) A complexidade espacial é S(n)=O(1).
B) A complexidade espacial é S(n)=O(log(n)).
C) A complexidade espacial é S(n)=O(n).
D) A complexidade espacial é S(n)=O(2).
E) Nenhuma das anteriores.

7. A pesquisa binária é tipicamente mais rápida do que a pesquisa sequencial.

A) A pesquisa binária só é mais rápida se o vetor não estiver ordenado caso contrário é melhor pesquisar sequencialmente.
B) A pesquisa binária só é mais rápida para vetores de grandes dimensões (mais de 1000 elementos).
C) Normalmente é duas vezes mais rápida pois divide em cada iteração o vetor em duas partes.
D) A pesquisa binária é sempre mais rápida mesmo para vetores desordenados.
E) Nenhuma das anteriores.

8. Qual é o método de ordenação preferível para vetores de média/grande dimensão?

A) Ordenação por Partição (QuickSort) pois tem complexidade temporal O(n^2);
B) Ordenação por Inserção pois tem complexidade temporal o(n*log n);
C) Ordenação por MergeSort pois tem uma complexidade espacial melhor que o Quicksort;

D) Ordenação por BubbleSort pois o processo de ordenação deste método é muito rápido;
E) Nenhuma das Anteriores

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

A) Com um método de escolha do Pivot que retorne a mediana do array, tem complexidade no tempo $O(n^2)$ e complexidade no espaço O(n).
B) No caso médio (valores aleatoriamente distribuídos) tem complexidade no tempo O(n*log(n)) e complexidade no espaço O(n*log(n)).
C) Com um método de escolha do Pivot que retorne a mediana do array, tem complexidade no tempo O(n*log(n)) e complexidade no espaço O(log(n)).
D) No caso médio tem uma complexidade no espaço de ordem inferior à ordenação por inserção pois ocupa menos memória.
E) Nenhuma das anteriores.

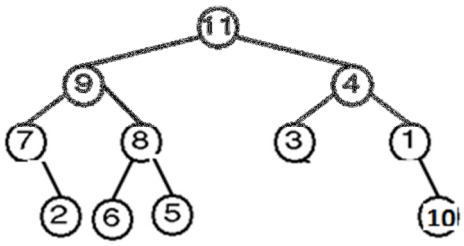
10. O método de ordenação por inserção:

A) Tem complexidade temporal O(n*log(n)) e espacial O(n).
B) Tem complexidade temporal O(n*log(n)) e espacial O(1).
C) Tem complexidade temporal $O(n^2)$ e espacial O(1).
D) Tem complexidade temporal $O(n^2)$ e espacial O(n).
E) Nenhuma das anteriores.

11. A classe *stack* 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 fila - *queue*).
B) Sim, mas só se se declarar no código fonte que *stack* é do tipo abstrato.
C) Não, pois o tipo de dados abstrato é a *queue*.
D) Sim, pois contém dados e operações sobre esses dados.
E) Não pois encapsula os dados e operações.

Suponha a seguinte árvore binária:



12. 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) 10 1 3 4 11 5 8 6 9 2 7
B) 10 1 3 4 5 6 8 2 7 9 11
C) 2 7 5 6 8 9 10 1 3 4 11
D) 5 6 8 2 7 9 10 1 3 4 11
E) Nenhuma das anteriores.

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

A) 2 7 6 5 8 9 11 3 10 1 4
B) 7 2 9 6 5 8 11 3 4 10 1
C) 7 2 9 6 8 5 11 3 4 1 10
D) 2 6 5 7 8 9 10 3 1 4 11
E) Nenhuma das anteriores.

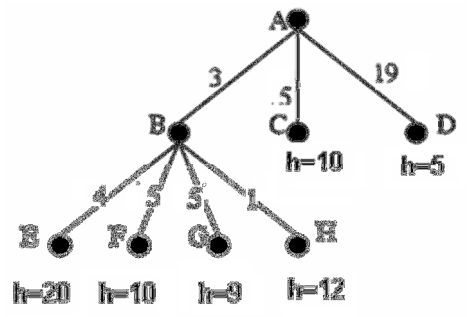
14. A operação de remoção de um elemento numa pilha

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

15. A operação de inserção 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.

16. 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 E ii) Nó E
E) Nenhuma das anteriores.

17. Diga quais os nós expandido em seguida utilizando respetivamente a Pesquisa Gulosa/Gananciosa e a pesquisa A*.

A) Gulosa: Nó D ; Algoritmo A*: Nó C
B) Gulosa: Nó D ; Algoritmo A*: Nó H
C) Gulosa: Nó C ; Algoritmo A*: Nó G
D) Gulosa: Nó C ; Algoritmo A*: Nó D
E) Nenhuma das anteriores.

18. Supondo o seguinte código, selecione a afirmação verdadeira.

```
Obj *ptr = new Obj[5];
delete ptr;
```

A) No final da execução o vetor é destruído (apagado).
B) O código dá erro de compilação
C) No final da execução só o primeiro elemento do vetor é destruído/apagado.
D) No final da execução nenhum elemento do vetor é destruído/apagado.
E) Nenhuma das respostas anteriores.

GRUPO II (8 Valores)

A biblioteca da Universidade do Minho organiza os seus livros por temas, mantendo essa informação num vetor (do tipo vetor). Cada elemento deste vetor é do tipo CTemaLivros (classe previamente definida). Esta classe contém como membros-dado uma string que identifica o tema (tema) e uma pilha com o título de todos os livros deste tema (livros). Ou seja, temos um vetor de temas em que cada tema tem um nome e uma pilha de títulos de livros.

```
class CTemaLivros {
    string tema;
    stack<string> livros;
  public:
    CTemaLivros(string tm="diversos"): tema(tm) {}
    string getTema();
    stack<string> &getLivros() { return livros; }
    bool operator!= (const CTemaLivros &l1) { return tema!=l1.tema; }
};
vector <CTemaLivros> bibUM;
```

2.1) Implemente na classe `CTemaLivros` a função:
`string getTema();`

2.2) Implemente a função que imprime o título de todos os temas existentes no vetor:
`void imprimeTemas(vector <CTemaLivros> bib);`

2.3) Implemente a função que ordena os títulos de todos os temas existentes no vetor:
`void ordenaTemas(vector <CTemaLivros> bib);`

2.4) Implemente a função que imprime o título de todos os livros existentes na biblioteca, apresentando-os por temas: `void escreveBib(vector<CTemaLivros> bib)`. Esta função escreve no monitor o título de cada tema, seguido dos títulos dos livros desse tema existentes na biblioteca bib sem alterar o conteúdo da biblioteca.

2.5) Implemente a função que procura na lista o elemento relativo a um tema especificado. `CTemaLivros procuraTema(const vector<CTemaLivros> bib, const string temaX)`. Esta função pesquisa a lista de temas da biblioteca bib, e retorna o elemento relativo ao tema `temaX` especificado como argumento.

2.6) Quando um livro é devolvido (ou um livro novo é comprado), este é colocado no topo da pilha correspondente ao seu tema. Implemente a função que realiza o processo de devolução (ou compra) de um livro: `void devolucao(vector<CTemaLivros> bib, const string nomeL, const string temaX)`. Esta função repõe na biblioteca `bib` o livro de título `nomeL`, pertencente ao tema `temaX`.

2.7) Os livros que estavam no topo das pilhas começam a ficar um pouco estragados, devido ao maior movimento que apresentam. Implemente a função que elimina o elemento do topo de todas as pilhas referentes a todos os temas que existem na biblioteca, `void eliminaTopo(vector<CTemaLivros> bib)`.

GRUPO III (5 Valores)

3.1) Implemente os seguintes membros-função para a classe *`Stack`* (pilha de inteiros) descrita nos slides/aulas da disciplina.

a) `void introduz_nX(int n, int x)` que adiciona n elementos com valor x no topo da pilha.

b) `int elimina_menor()` que elimina o menor dos três elementos que se encontram no topo da pilha. Devem ser considerados os casos em que a pilha tem menos de três elementos ou está mesmo vazia. A função deve retornar o elemento eliminado ou 0 caso a pilha esteja vazia.

3.2) Suponha a classe `List` descrita nos slides/aulas da disciplina.

a) Implemente o método `int contaEntre(int x1, int x2);` que conte o número de elementos de uma lista que são maiores que x1 e menores que x2. A função deve escrever no écran todos os elementos nestas condições e retornar a sua contagem. Exemplo: Aplicar `contaEntre(2,8)` a [1 10 5 8 2 2 3 4 5 9] resulta na escrita de 5 3 4 5 e é retornado o valor 4.

b) Construa uma função `int zipar(int x1, int x2);` que elimine todos os elementos consecutivos iguais de uma lista ordenada retornando o número de elementos eliminados. Por exemplo: aplicando `zipar` a [1 1 2 2 2 3 3 3 3 4 4 7 8 8] resulta em [1 2 3 4 7 8] e é retornado o valor 8.

3.3) Na classe BTree descrita nos slides/aulas da disciplina implemente o método:

```
void escreve_maiores(ostream &os);
```

Este método percorre a árvore e determina quais os dois elementos maiores presentes na árvore e escreve no output stream considerado, o número total de elementos existente na árvore e os dois maiores elementos ordenados por ordem decrescente. Devem ser considerados os casos em que a árvore está vazia ou só tem um elemento. Exemplos:

```
A árvore tem 8 elementos e os maiores são 25 e 18.

A árvore está vazia.

A árvore tem 1 único elemento com valor 23.
```