

**GRUPO 1 (3.0 Val.)**

1. O código que se apresenta abaixo é uma alteração mínima a um dos programas fornecidos como exemplo à matéria sobre listas de objectos.

```
1.  #include <iostream>
2.  #include <string>
3.  #include <vector>
4.  using namespace::std;
5.
6.  bool procurar (const vector<string> &v, string s);
7.
8.  void iniciar(vector<string> &v)
9.  { v.push_back("César Augusto"); }
10.
11. void listar(const vector<string> &v)
12. {
13.     vector<string>::iterator it;
14.     for (it = v.begin(); it != v.end(); it++)
15.     { cout << "\n" << i << ":\t" << *it; }
16.     cout << "\n";
17. }
18.
19. int main()
20. {
21.     vector<string> nomesP2;
22.     cout << "Nº de nomes: " << nomesP2.size() << endl;
23.     iniciar(nomesP2);
24.     listar (nomesP2);
25.     if (procurar(nomesP2,"Carlos Magno"))
26.         cout << "\nEncontrado";
27.     else
28.         cout << "\nNão encontrado";
29.     return 0;
30. }
```

- Identifique e reproduza a linha em que se define a lista de objectos.
- Pretende-se alterar o código por forma a se utilizar a *container class list* da STL. Identifique todas as linhas em que é necessário efectuar alterações e apresente novas linhas, de substituição.
- Escreva o código da função `procurar()` declarada na linha 6.

GRUPO 2 (3.0 Val.)

2. Em problemas das aulas práticas, surgiram casos de simulação de actividades comerciais em que o atendimento de clientes era efectuado por via de duas filas, uma prioritária e outra normal, tal como exemplificado no seguinte extracto de código.

```
1.  class Loja
2.  {
3.      queue<Cliente> prioritaria;
4.      queue<Cliente> normal;
5.      /*...*/
6.      void insereCliente (Cliente cl, bool tipoCliente);
7.      Cliente proximoCliente(queue<Cliente> & fila);
8.      void serveCliente(Cliente cl);
9.      int numeroClientes(const queue<Cliente> & fila);
10.     /*...*/
11. };
```

Na loja aqui representada, as diferentes filas são atendidas por diferentes funcionários, mas quando um deles tem de sair mais cedo, as filas juntam-se numa fila única respeitando as prioridades dos clientes em espera.

Usando as funcionalidades das classes modelo de STL, escreva, da forma o mais simples possível (por exemplo, sem preocupações de eficiência de código) uma função-membro

```
void Loja::juntarFilas()
```

para cada uma das seguintes situações:

- a. todos os clientes normais juntam-se à fila com clientes prioritários;
- b. todos os clientes prioritários juntam-se à fila com clientes normais, mantendo a primazia de atendimento.

GRUPO 3 (3.0 Val.)

3. Pretende-se implementar um sistema para armazenamento e consulta de telefones de assinantes de uma determinada zona. Neste sistema a pesquisa de informação é realizada sobre o nome do assinante, isto é, pretende-se obter o número de telefone e morada de um assinante a partir do seu nome. A classe **Assinante**, representada a seguir de forma incompleta, contém a informação relativa a um assinante:

```
class Assinante {
    string telefone, nome, morada;
public:
    Assinante(string tf, string n, string m):
        telefone(tf), nome(n), morada(m) {};
    ...
};
```

A informação sobre os assinantes é guardada numa árvore binária de pesquisa (**BST**).

- a. Represente graficamente a árvore resultante da inserção, pela ordem indicada, dos seguintes assinantes:

```
("223542676", "Manuel Santos", "R.Camoes")
("226782599", "Rita Silva", "R.Camoes")
("229842311", "Filipe Costa", "R.Camoes")
("227345283", "Paulo Lemos", "R.Camoes")
("225437864", "Ines Sousa", "R.Camoes")
```

Justifique a sua resposta.

- b. Implemente a função *procuraNome*, que efectua a pesquisa de um determinado assinante através do seu nome (*nomeX*) passado como argumento, e imprime no monitor a morada e nº de telefone:

```
void procuraNome(const BST<Assinante> &arv, string &nomeX)
```

Se não existir um assinante com nome igual a *nomeX*, deve ser escrita no monitor a informação relativa a todos os assinantes cujo nome se inicia com o primeiro carácter de *nomeX*.

GRUPO 4 (3.0 Val.)

4. Considere de novo o enunciado do grupo anterior, mas agora pretende-se que a pesquisa de informação seja realizada sobre o nº de telefone do assinante, isto é, pretende-se obter o nome e morada de um assinante a partir do seu nº de telefone. Para isso, usa-se uma tabela de dispersão:

```
typedef hash_set<Assinante, HAssin, IgAssin> HTAssinante;
```

- a. Implemente as funções de dispersão e igualdade requeridas pelo uso da classe **hash_set**.
- b. Implemente a função que cria uma tabela de dispersão a partir de uma árvore binária de pesquisa passada como argumento, e que contém um conjunto de assinantes:

```
HTAssinante criaTabela(const BST<Assinante> &arv)
```

GRUPO 5 (8.0 Val.)

Responda às seguintes questões, preenchendo a tabela com a opção correcta (em maiúsculas). Cada resposta certa vale 1 ponto; cada resposta errada vale - 0,3 pontos (note o sinal menos!). O total é 20 pontos (8 valores da nota final da prova).

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

- 5.1) Um **iterador** deve ser sempre associado a estruturas de dados lineares.
- Sim, pois só assim se poderá aceder aos elementos dessa estrutura
 - Talvez, desde que se trate de facilitar o acesso aos elementos dessa estrutura
 - Não, pois há situações em que não faz sentido usar iteradores
 - Nenhuma das respostas anteriores
- 5.2) A classe **list** da STL inclui um método (**sort**) que permite ordenar os elementos de uma lista instanciada. Mas tais elementos também poderiam ser ordenados usando o algoritmo genérico **sort**, da STL.
- Sim, pois o resultado seria o mesmo
 - Talvez, mas só se a eficiência não for importante
 - Não, por causa do tipo de iteradores usados em `list`
 - Nenhuma das respostas anteriores
- 5.3) Na classe **stack** da STL a operação `pop()` está declarada como retornando `void`. Mas poderia, com vantagem, retornar «`value_type &`», tal como acontece com `top()`.
- Não, porque perderia eficiência
 - Não, porque diminuiria a utilização de `top()`
 - Não, porque originaria um erro grave
 - Nenhuma das respostas anteriores
- 5.4) A escolha entre a utilização num programa de bibliotecas dinâmicas ou bibliotecas estáticas
- Deve ser deixada ao critério do utilizador
 - É completamente irrelevante, desde que se garanta que o programa funciona.
 - Deve ser ponderada, pois pode afectar utilização de memória do sistema
 - Nenhuma das respostas anteriores
- 5.5) Para se inserir um elemento novo numa **lista de objectos**, pode ser necessário copiar e deslocar elementos se a implementação utilizada for baseada em:
- `Array`
 - Lista ligada
 - Lista duplamente ligada
 - Nenhuma das respostas anteriores
- 5.6) A operação de depuração (*debugging*) de um programa é normalmente feita:
- Pelo programador, utilizando, ou não, um depurador (debugger)
 - Pelo programador, utilizando, ou não, um interpretador
 - Pelo utilizador, utilizando ou não, um depurador (debugger)
 - Pelo utilizador, utilizando, ou não, um interpretador
- 5.7) Todos os métodos `push()`, `pop()`, `empty()` e `size()` costumam ser disponibilizados em:
- Pilhas
 - Pilhas e filas
 - Pilhas, filas e listas
 - Nenhuma das respostas anteriores
- 5.8) Um **programa** típico em C++, para ser utilizado, deve ser, sequencialmente:
- Editado, compilado e executado
 - Compilado, executado e editado
 - Interpretado, compilado e executado
 - Editado, interpretado e executado
- 5.9) O que se entende por “**código-fonte**” de um programa?
- É o código que identifica o programa e as plataformas onde pode ser utilizado
 - É o código que foi escrito por quem desenvolveu o programa
 - É o código que está preparado para ser executado na máquina onde for colocado

- D) Nenhuma das respostas anteriores
- 5.10) Uma **lista de objectos** permite armazenar uma sequência de objectos do mesmo tipo.
- A) Sim, é esse o sentido tradicional de lista
 - B) Talvez, desde que não haja objectos exactamente iguais
 - C) Não, pois uma lista pode conter objectos de tipos diferentes
 - D) Nenhuma das respostas anteriores
- 5.11) Numa árvore binária:
- A) Qualquer nó tem tamanho superior a qualquer dos seus filhos
 - B) Qualquer nó tem tamanho superior ao do seu pai
 - C) Um nó não folha tem pelo menos um filho de tamanho superior
 - D) Nenhuma das respostas anteriores
- 5.12) Uma **árvore binária** de altura 3:
- A) Não pode ter mais de 8 folhas
 - B) Não pode ter mais de 4 folhas
 - C) Tem sempre 4 ou mais folhas
 - D) Nenhuma das respostas anteriores
- 5.13) Conhecendo apenas a sequência de valores que representa a visita de uma qualquer **árvore binária** é possível reconstruir essa mesma árvore:
- A) Não é possível reconstruir a mesma árvore
 - B) Se a sequência de valores representa a visita em pós-ordem da árvore binária
 - C) Se a sequência de valores representa a visita em pré-ordem da árvore binária
 - D) Nenhuma das respostas anteriores
- 5.14) Numa **árvore binária de pesquisa**, a ordem de inserção dos elementos:
- A) Não influencia a visita pós-ordem da árvore
 - B) Não influencia a visita pré-ordem da árvore
 - C) Não influencia a visita em-ordem da árvore
 - D) Nenhuma das respostas anteriores
- 5.15) Numa **árvore binária de pesquisa**, o caminho entre a raiz e um qualquer nó é sempre:
- A) Uma sequência de valores decrescentes
 - B) Uma sequência de valores crescentes
 - C) Uma sequência de valores alternadamente crescentes e decrescentes
 - D) Nenhuma das respostas anteriores
- 5.16) Numa **árvore binária de pesquisa**, é obtida uma sequência de valores crescentes:
- A) Na visita em ordem
 - B) Na visita em pré-ordem
 - C) Na visita em pós-ordem
 - D) Nenhuma das respostas anteriores
- 5.17) Numa **tabela de dispersão** com resolução de colisões por listas, o factor de carga:
- A) É sempre inferior a 0.5
 - B) É sempre superior a 1
 - C) Pode ser superior a 1
 - D) Nenhuma das respostas anteriores
- 5.18) Numa **tabela de dispersão**, elementos com a mesma chave:
- A) São sempre elementos iguais (repetidos)
 - B) São sempre elementos diferentes
 - C) Não existem elementos com a mesma chave
 - D) Nenhuma das respostas anteriores
- 5.19) O vector [12, 5, 9, 3, 4, 5] representa:
- A) Uma fila de prioridade de máximo
 - B) Uma fila de prioridade de mínimo
 - C) Não representa uma fila de prioridade, porque a fila de prioridade não admite elementos repetidos
 - D) Nenhuma das respostas anteriores
- 5.20) Na **fila de prioridade** representada pelo vector [20, 15, 10, 7, 2, 8], a remoção do maior elemento origina:
- A) A fila de prioridade representada pelo vector [15, 8, 10, 7, 2]
 - B) A fila de prioridade representada pelo vector [10, 15, 8, 7, 2]
 - C) A fila de prioridade representada pelo vector [15, 7, 10, 8, 2]
 - D) Nenhuma das respostas anteriores