

1. Pretende-se fazer a gestão informática das candidaturas dos alunos a um curso de mestrado, de tal modo que para cada candidatura é recolhido o nome do candidato, licenciatura, média da licenciatura, número de anos de actividade profissional e área de estudo pretendida. As candidaturas são colocadas numa pilha. Após o período de candidaturas estas são processadas sendo colocadas na lista relativa à área pretendida, por ordem decrescente de média de curso e anos de profissão. Admita que existem três áreas de especialização e utilize um vector de listas para guardar as candidaturas às três áreas de especialização do mestrado. Considere a classe **FichCandid** que guarda informação relativa a um candidato:

```
class FichCandid
{
    private:
        string area;
        string nome;
        string licenc;
        double media;
        int anosprofiss;

    public:
        FichCandid ();
        FichCandid (string n, string l, double m, int a);
        FichCandid (const FichCandid& fc);
        ~FichCandid () ;

        string getArea() const ;
        string getNome() const ;
        string getLic() const ;
        double getMedia() const ;
        int getAnosProfiss()const ;

        void setArea (string area) ;
        void setNome (string nom) ;
        void setLic (string lic) ;
        void setMedia (double m) ;
        void setAnosProfiss (int a) ;

        FichCandid& operator = (const FichCandid& fc) ;
        bool operator == (const FichCandid& fc) const ;
        void escreve (ostream& out) const ;
};
```

- a) Defina a classe **GestCand** para gerir as candidaturas a este mestrado. Indique apenas os atributos e a assinatura dos métodos necessários ao bom funcionamento do sistema de gestão de candidaturas acima descrito.
- b) Implemente o método **gereCandiduras** que separa os candidatos por área pretendida e por ordem decrescente de média de curso e anos de profissão.
- c) Faça um método recursivo para listar o vector de candidaturas.

2. Considere o seguinte método:

```
template <class T>
const Lista<T>& Lista<T>::operator ~ ()
{
    int compr = comprimento() ;
    T aux1, aux2 ;

    int j = compr ;
    for (int i = 1 ; i < j ; i++)
    {
        if (i != j)
        {
            encontra(i,aux1) ;
            encontra(j,aux2) ;

            insere(i,aux2) ;
            remove(i+1,aux2);

            insere(j,aux1) ;
            remove(j+1,aux1);
        }
        j-- ;
    }
    return *this;
}
```

- a) Explique o que faz o operador acima codificado.
- b) Analise a sua complexidade temporal, notação BigOh. Justifique.
3. Pretende-se representar as várias zonas de um Centro Comercial, "Restauração", "Desporto", "Calçado", "Parque Estacionamento Nascente", etc... e os tipos de ligação entre as zonas usando um grafo dirigido, implementado através da classe template ListAdjGrafo<TV,TR>. As várias zonas do Centro Comercial podem estar ligadas por "corredor pedestre", "escada rolante", "passadeira rolante" ou "elevador" e sabe-se o tempo de ligação entre as mesmas. Podem existir vários tipos de ligação entre duas zonas.
- a) Apresente a definição das classes necessárias à modelação deste problema (não é necessário indicar os métodos das classes).
- b) Implemente o método **LigsZona** que para uma determinada zona indica os diferentes tipos ligação que a servem.
- c) Implemente o método **ZonaCentral** que para uma determinada zona verifica se esta tem ligações directas com todas as restantes zonas.

4. Considere um sistema de rega disposto em forma de árvore binária em que a quantidade de água disponibilizada no ponto inicial vai repartir-se por todos os canais de tal modo que, quando há uma bifurcação, a quantidade de água é dividida de igual forma por cada um dos canais. Adicione à classe template `ArvBinPesq<T>` um método que apresente a quantidade de água que chega a cada um dos nós folha.