1. Pretende-se fazer a gestão informática das inscrições dos alunos de uma escola, de tal modo que as fichas de inscrição de cada aluno, com o seu nome, data de nascimento e área de estudo pretendida, são colocadas numa fila de espera, por ordem de chegada. Após o término das inscrições estas são processadas, atribuindo a cada aluno uma turma e um número de acordo com a área de estudo pretendida, a idade e a ordem de inscrição. Cada turma só pode ter no máximo 20 alunos da mesma área de estudo. Utilize um vector dinâmico de listas para guardar as várias turmas e considere a classe **FichAluno** que guarda informação relativa a um aluno:

```
class FichAluno
{
    private:
       string nome;
       string area;
       Data dtnasc;

    public:
       FichAluno ();
       FichAluno (string m, string a, Data d);
       FichAluno (const FichAluno& fa);
       ~FichAluno () ;

       string getNome() const ;
       string getArea() const ;
       int getIdade() const ;

       void setNome (string nom) ;
       void setArea (string area) ;
       void setDtNasc (const Data& dt) ;


       FichAluno& operator = (const FichAluno& fa) ;
       bool operator == (const FichAluno& fa) const ;
       void escreve (ostream& out) const ;
};
```


a) Defina a classe **GestInscr** para gerir as fichas de inscrição e as turmas desta escola. Indique apenas os atributos e a assinatura dos métodos necessários ao bom funcionamento do sistema de inscrições acima descrito.

b) Implemente o método **criaTurmas** que cria as turmas necessárias para processar a fila de espera de inscrições, colocando os alunos nas turmas por área pretendida, ordem crescente de idade e para os alunos da mesma idade, considera a ordem de inscrição.

c) Faça um método recursivo para listar todas as turmas.

2. Considere o seguinte método que faz a sobrecarga de um operador na classe Queue:

```
template<class T>
void Queue<T>::operator % (Queue<T> &q)
{
    int comp = comprimento(), i=1 ;
    T elem ;
    Queue<T> aux ;

    while (i <= comp)
    {
       retira(elem);
       if (i <= comp/2)
          aux.insere(elem) ;
       else
          q.insere(elem) ;
       i++ ;                }
    while (!aux.vazia())
    {
       aux.retira(elem);
       insere(elem) ;     }
}
```


a) Explique o que faz o operador acima codificado.

b) Analise a sua complexidade temporal, notação BigOh. Justifique.

3. Uma determinada cidade encontra-se dividida em locais de interesse turístico ligados por caminhos pietonais. Cada um dos locais caracteriza-se por uma designação e pelo tempo médio de visita ao local. As ligações directas entre os diferentes locais referem também o tempo médio que leva a percorrê-las a pé. Supondo que esta rede de locais turísticos é representada por um grafo dirigido da classe template ListAdjGrafo<TV,TR>:

a) Esboce a declaração das classes que dão suporte à representação desta rede de locais turísticos, indicando claramente os atributos das classes (não é necessário indicar os métodos das classes)

Desenvolva os métodos que permitam:

b) Apresentar o tempo médio de visita total de todos os locais de interesse turístico

c) Dado um local turístico inicial apresentar o local turístico que se consegue visitar mais rapidamente.

4. Adicione à classe template ArvBinPesq<T> o método que imprime os elementos da árvore contidos num intervalo de valores.