Nota: Submeta a pasta contendo todos os ficheiros da sua resolução comprimida num único ficheiro *zip*.

GRUPO 1

Um grupo nacional detentor de várias pistas de Karts quer fazer uma aplicação muito simples para assegurar a gestão das suas pistas e da sua frota atual de viaturas. Cada pista, aqui representada por `CPista` tem uma frota associada de Karts, class `CKart`, utilizando para isso um `vector` de karts `vector<CKarts> frotaKartsPista`. Para isso, propõe-se a utilização da seguinte estrutura de classes contendo as classes `CGrupo`, `CKart` **e** `CPista`, que estão parcialmente definidas a seguir:

```
class CKart{
  string nome;
  float cilindrada;
  int numero;
  bool avariado;
public:
  CKart(bool avar,string nom,int num,float cilind);
  string getNome();
  bool getAvariado();
  float getCilindrada();
  int getNumero();
};

class CPista{
  string nomePista;
  vector <CKart> frotaKartsPista;
  queue <CKart> kartsLinhaPartida;
  vector<CKart> kartsEmProva;
public:
  CPista(string nomePista, vector <CKart> frotaKarts);
  string getNome();
  vector <CKart> &getFrotaActual();
  vector <CKart> getKartsAvariados();
  bool prepararCorrida(int numeroKarts, int cilind); //Ex 1c)
  int inicioCorrida();  //Ex 1d)
};

class CGrupo{
  vector <CPista> pistasG;
public:
  vector <CPista> getPistas();
  void adicionaPista(CPista pista);
  vector <CKart> ordenaKarts();//Ex 1a)
  int numAvariados(int cilind); //Ex 1b)
};
```

a) Na classe CGrupo implemente a função que ordena pelo seu numero e devolve um `vector` da frota de Karts (disponíveis em todas as pistas e em qualquer estado):

```
vector<CKart> CGrupo::ordenaKarts();
```

b) Na classe CGrupo implemente uma função que retorne o número total de Karts avariados com uma dada cilindrada para todas as pistas:

```
int CGrupo::numAvariados(int cilind);
```

Nota: Possui já uma função que lhe fornece um vetor com os carros avariados de uma dada pista (`vector <CKart> CPista::getKartsAvariados()`)

c) Quando uma nova corrida se vai iniciar numa pista, um conjunto de karts da mesma cilindrada e que não estejam avariados é selecionado da frota de Karts (`vector<CKarts> frotaKartsPista`) e colocado na fila que representa a linha de partida (`queue<CKarts> kartsLinhaPartida`). Implemente na classe CPista o método:
`bool CPista::prepararCorrida(int numeroKarts, int cilind);`
de forma a que selecione pela mesma ordem em que estão na frota de carros o numero de karts `numeroKarts`, funcionais e com cilindrada `cilind`, a colocar em prova, e devolva uma fila com os carros respetivos. Caso não existam Karts suficientes funcionais com essa cilindrada são colocados em prova todos os disponíveis e a função retorna false.

d) Simule o início de uma prova, implementando o método:
`int CPista::inicioCorrida();`
Este método deve retirar os karts da Fila de partida e colocar esses karts, pela mesma ordem, no `vector kartsEmProva`, retornando o número de Karts em prova

GRUPO 2

Implemente os seguintes membros-função para a classe `CStack` (pilha de inteiros) fornecida.

a) `void CStack::adicionaN (int n);`
que verifica qual o valor v do elemento no topo de uma pilha e adiciona n elementos com valores {v+1, v+2, … v+n} sucessivamente no topo da pilha. Exemplo: Aplicando adicionaN(4) a uma pilha com três elementos { 3 2 1 } passa a { 7 6 5 4 3 2 1 }.

b) `bool CStack::inverte4();`
que inverte os 4 elementos do topo da pilha. Exemplo: { 5 4 3 2 1 } passa a {2 3 4 5 1}. Caso a pilha tenha menos de 4 elementos a função não faz nada e retorna false.

Implemente os seguintes membros-função para a classe `CSimpleList` (lista simples) fornecida.

c) `void CSimpleList::intercalar(const CSimpleList &lst)`
O método serve para intercalar numa `CSimpleList` os elementos de outra `CSimpleList` passada como parâmetro. A segunda lista não é modificada. Exemplo: [1 2 3 4 5] intercalada com [6 7 8] passaria a conter [1 6 2 7 3 8 4 5].

d) `int CSimpleList::zipar();`
O método deve eliminar todos os elementos consecutivos iguais de uma `CSimpleList` ordenada retornando o número de elementos eliminados. Por exemplo: aplicando a função zipar a uma lista com [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.