9ª aula prática – Tabelas de dispersão

- Faça download do ficheiro ***aed2223_p09.zip*** da página e descomprima-o o (contém a pasta ***lib***, a pasta ***Tests*** com os ficheiros ***funHashingProblem.h, funHashingProblem.cpp, jackpot.h, jackpot.cpp*** e ***tests.cpp***, e os ficheiros ***CMakeLists*** e ***main.cpp***)
- No CLion, abra um ***projeto***, selecionando a pasta que contém os ficheiros do ponto anterior.

1. Considere a classe **FunHashingProblem** que possui apenas métodos estáticos. Implemente a função:

```
vector<int> FunHashingProblem::findDuplicates(vector<int> values, int k)
```

Dado um vector de números inteiros *values* e um número positivo *k*, pretende-se verificar se o vector contém algum elemento duplicado dentro do intervalo ***k***. Se ***k*** for maior do que o tamanho do vector, a solução deve verificar se há elementos duplicados em todo o vector. A função devolve um vector com os elementos duplicados (pela ordem em que os encontra) ou vazio, caso não exista nenhum.
Sugestão: use uma tabela de dispersão (**unordered_set**) para guardar os elementos que estão no intervalo *k*.

Complexidade temporal esperada: $O(n)$

Exemplos de execução:

input: `values = {5, 6, 8, 2, 4, 6, 9} e k = 4`

output: {6}

A solução encontra o elemento 6 duplicado, que está repetido a uma distância 4 que é <= k.

input: `values = {5, 6, 8, 2, 4, 6, 9} e k = 2`

output: {}

Não encontra duplicados: o elemento 6 está duplicado, mas está repetido a uma distância > k.

input: `values = {1, 2, 3, 2, 1} e k = 7`

output: {1, 2}

O elemento 1 duplicado está repetido a uma distância 4; o elemento 2 duplicado a uma distância 2, ambas <= k.

2. Pretende-se implementar um programa para ajudar na gestão de apostas no jogo totoloto. Considere a classe **Jackpot** que guarda as apostas existentes numa tabela de dispersão.

nota: deve realizar este exercício respeitando a ordem das alíneas.

```
class Bet
{
    vector<int> numbers;
    string player;
    //…
public:
    Bet(vector<int> ns, string p);

};

typedef unordered_set<Bet, betHash, betHash> tabHBet;
class Jackpot
{
    tabHBet bets;
public:
    void addBet(const Bet& b);
    unsigned betsInNumber(unsigned num) const;
    tabHBet drawnBets(const tabHInt& draw) const;
};
```

a) Implemente o membro-função:

```
void Jackpot::addBet(const Bet& b)
```

Esta função acrescenta uma dada aposta *b* ao conjunto de apostas existentes. Considere que um jogador não pode efetuar apostas iguais.

b) Implemente o membro-função:

```
unsigned Jackpot::betsInNumber(unsigned num) const
```

Esta função determina quantas vezes aparece o número *num,* no total de apostas efetuadas.

c) Implemente o membro-função:

```
vetor<string> Jackpot::drawnBets(vector<int> draw) const
```

Esta função retorna um vetor com o nome dos jogadores com apostas premiadas (isto é, aquelas que possuem mais de 3 valores iguais aos valores de sorteio.