Practical 9 – Hash Tables

- Download the ***aeda2021_p09.zip*** file from the moodle page and unzip it (contains the ***lib*** folder, the ***Tests*** folder with files ***bet.h, bet.cpp, player.h*** and ***player.cpp*** and ***tests.cpp*** and the ***CMakeLists*** and ***main.cpp*** files)

You must perform the exercise respecting the order of the questions

Exercise

1. We want to implement a program that generates several bets and compares them with the numbers drawn. For this purpose, consider the **Bet** class that stores the numbers of a bet in a hash table (**unordered_set**). A bet can have between 6 and 12 numbers.

```
typedef unordered_set<unsigned> tabHInt;

class Bet
{
    tabHInt numbers;
public:
    Bet() {};
    void generateBet(const vector<unsigned> &values, unsigned n=6);
    bool contains(unsigned num) const;
    unsigned countRights(const tabHInt& draw) const;
    tabHInt getNumbers() const { return numbers; }
};
```

a) Implement the following member function in class **Bet**:

```
void Bet::generateBet(const vector<unsigned> &values, unsigned n)
```

Consider *values* a vector of m (m> n) randomly generated values. This function creates a bet that includes the first *n* unrepeated numbers from the *values* vector and stores those numbers in the member data (table) *numbers*.

b) Implement the following member function in class **Bet**:

```
bool Bet:contains(unsigned num) const
```

This function determines whether the given number *num* is contained in the bet.

c) Implement the following member function in class **Bet**:

```
unsigned Bet::countRights(const tabHInt& draw) const
```

This function counts the correct numbers in the bet, according to the *draw*.

2) Following the previous exercise, now consider the Player class that holds multiples bets from a player.

```
typedef unordered_set<Bet, betHash, betHash> tabHBet;
class Player
{
    tabHBet bets;
    string name;
public:
    Player(string nm = "anonymous") { name=nm; }
    void addBet(const Bet &b);
    unsigned betsInNumber(unsigned num) const;
    tabHBet drawnBets(const tabHInt& draw) const;
   unsigned getNumBets() const { return bets.size(); }
};
```

a) Implement the following member function in class **Player**:

```
void Player::addBest(const Best& b)
```

This function adds a given bet *b* to the player's set of bets.

b) Implement the following member function in class **Player**:

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

This function determines how many times the player has bet on a given number *num*, in the total bets made.

c) Implement the following member function in class **Player**:

```
tabHBet Player::drawnBets(const tabHInt& draw) const
```

This function returns a hash table containing the player's bets that are awarded (that is, those that have more than 3 values equal to the draw values).