Practical 5 - Lists

Instructions

- Download the ***aeda2021_p05.zip*** file from the moodle page and unzip it (contains the ***lib*** folder, the ***Tests*** folder with files ***kid.h, kid.cpp***, ***game.h, game.cpp*** and ***tests.cpp*** and the ***CMakeLists*** and ***main.cpp*** files)
- You must perform the exercise respecting the order of the questions

Exercise

"*Pim Pam Pum cada bola mata um pra galinha e pro peru quem se livra és mesmo tu*". Remember this children's game, which rules are as follows:

The first child says the sentence, and in each word he points to each of the children at play (starting with himself). When it reaches the end of the list of children, it returns to the beginning, that is, itself.

- The child being pointed out, when the last word of the sentence is said, gets free and leaves the game. Counting resumes at the next child.
- The child who remains loses the game.

Use a list (use the STL **list** class) to implement this game. The list elements are objects of the **Kid** class, defined below (see file *kid.h*) :

```
class Kid {
    string name;
    unsigned age;
public:
    Kid();
    Kid(string nm, unsigned a);
    Kid(const Kid &k1);
    unsigned getAge() const;
    string getName() const;
    string write() const;
};
```

The class **Game** is defined as:

```
class Game
{
    list<Kid> kids;
public:
    Game();
    Game(list<Kid>& l2);
     static unsigned numberOfWords(string phrase);
    void addKid(const Kid &k1);
    list<Crianca> getKids() const;
    string write() const;
    Kid& loseGame(string phrase);
    list<Kid>& reverse();
    list<Kid> removeOlder(unsigned a);
    void setKids(const list<Kid>& l2);
    bool operator==(Game& g2);
    list<Kid> shuffle();
};
```

a) Implement **Game** class constructors and member functions:

```
void Game::addKid(const Kid &k1)
```

This function adds kid *k1* to the game.

```
list<Kid> Game::getKids() const
```

This function returns the list of kids currently at game.

```
void Game::setKids(const list<Kid> &l1)
```

This function sets the list of kids l1 at game.

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

```
string Game::write() const
```

This function returns a string with all the kids that are playing at any given time. The information about each kid must be in the format "name : age" (see member-function *write ()* of the *Kid* class).

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

```
Kid& Game::loseGame(string phrase)
```

This function performs the enunciated game when the phrase used is *phrase* and returns the kid who loses the game. Use the member function *numberOfWords* (already provided) that counts the number of words of a phrase specified in parameter:

```
int Game::numberOfWords(string phrase)
```

d) Implement the following member function in class **Game**:

```
list<Kid>& Game::reverse()
```

This function inverts the order of kids/players in the *kids* list. Returns the reference to this list of kids.

e) Implement the following member function in class **Game**:

```
list<Kid> Game::removeOlder(unsigned a)
```

This function removes, from the game, kids older than the value specified as a parameter *a*. Returns a new list with the kids that have been removed.

f) Implement the equality operator in class **Game**:

```
bool Game::operator==(Game& g2)
```

Two games are considered equal if they have the same kids in their respective lists, in the same order.

g) Implement the following member function in class **Game**:

```
list<Kid> Game::shuffle() const
```

This function creates a new list where the kids in the game are placed in a randomly position (use the *rand ()* function to generate random numbers).