5ªaula prática - Listas

Faça download do ficheiro ***aeda2021_p05.zip*** da página da disciplina e descomprima-o (contém a pasta ***lib,*** a pasta ***Tests*** com os ficheiros ***kid.h, kid.cpp***, ***game.h, game.cpp*** e ***tests.cpp***, e os ficheiros ***CMakeLists*** e ***main.cpp***)

- Deverá realizar a ficha respeitando a ordem das alíneas.

Enunciado

"*Pim Pam Pum cada bola mata um pra galinha e pro peru quem se livra és mesmo tu*". Recorde este jogo de crianças, cujas regras são as seguintes:

- A primeira criança diz a frase, e em cada palavra vai apontando para cada uma das crianças em jogo (começando em si). Ao chegar ao fim da lista de crianças, volta ao início, ou seja, a ela mesma.
- A criança que está a ser apontada, quando é dita a última palavra da frase, livra-se e sai do jogo. A contagem recomeça na próxima criança.
- Perde o jogo a criança que restar.

Use uma lista (pode usar a classe *list* da STL) para implementar este jogo. Os elementos da lista são objetos da classe **Kid**, definida abaixo (ver ficheiro *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;
};
```

A classe **Game** também está declarada:

```
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) Implemente os construtores da classe **Game** e ainda os membros-função:

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

Esta função adiciona a criança *k1* ao jogo.

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

Esta função retorna a lista de crianças atualmente em jogo.

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

Esta função coloca em jogo a lista de crianças l1.

b) Implemente o membro-função da classe **Game**:

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

Esta função retorna uma *string* com todas as crianças que estão em jogo num dado momento. A informação sobre cada criança deve estar no formato "nome : idade" (ver membro-dado `write()` da classe `Kid`).

c) Implemente o membro-função da classe **Game**:

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

Esta função realiza o jogo enunciado quando a frase utilizada é *phrase* e retorna a criança que perde o jogo. Use o membro-função *numberOfWords* (já fornecido) que determina o número de palavras existentes numa frase indicada em parâmetro:

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

d) Implemente o membro-função da classe **Game**:

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

Esta função inverte a ordem das crianças/jogadores na lista *kids*, em relação à ordem original, e retorna a referência para a lista de crianças.

e) Implemente o membro-função da classe **Game**:

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

Esta função remove do jogo as crianças de idade superior ao valor *a* especificado como parâmetro, e retorna uma nova lista com as crianças que foram removidas.

f) Implemente o operador de igualdade para a classe **Game**:

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

Dois jogos são considerados iguais se possuem nas suas respetivas listas as mesmas crianças, na mesma ordem.

g) Implemente o membro-função da classe **Game**:

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

Esta função cria uma nova lista onde as crianças do jogo são colocadas em uma posição determinada aleatoriamente (use a função *rand()* para gerar números aleatórios).