```
/*****************************************************************************
*
*  Author:           Miklos Moreno
*  Email:            miklosam1999@gmail.com
*  Label:            P03-B
*  Title:            BolNiverse Fight Club
*  Course:           CMPS 2143
*  Semester:         Fall 2021
*
*  Description:
*        This program is a rolled back take on the BolNiverse Fight CLub
*        Using runtime polymorphism, the goal is to make attackers and
*        defenders fight eachother using unique weapons that deal a
*        random damage roll based on their weapon and 'luck'.
*
*  Usage:
*        $ ./main
*
*  Files:
*         main.cpp        : driver program
*         dice.hpp        : dice class for number rolls
*         fighters.hpp    : parent and child classes of type fighters
*         helpers.hpp     : aids for dice class and damage parameters
*         weapons.hpp     : parent and child classes of type weapons
*****************************************************************************/

#include <iomanip>
#include <thread>
#include <cstdlib>
#include <math.h>
#include "fighters.hpp"

using namespace std;

/*
*@brief: fills attacker vector
*@param: vector of attacker* from input
*/
void populateAttackers(vector<BaseFighter*>& attackers) {

  cout << "How many attackers do you wish to send? ";
  int raiders;
  cin >> raiders;

  for(int i = 0; i < (raiders) / 5; i++) {
    BaseFighter* archer = new Archer();
    BaseFighter* warrior = new Warrior();
    BaseFighter* wizard = new Wizard();
    BaseFighter* elf = new Elf();
    BaseFighter* dragonborn = new DragonBorn();
    attackers.push_back(archer);
    attackers.push_back(warrior);
```

```
    attackers.push_back(wizard);
    attackers.push_back(elf);
    attackers.push_back(dragonborn);
    }
}
/*
*@brief: fills defender vector based on amount of attackers
*@param: size of attackers, vector defender*
*/
void populateDefenders(int attackersSize, vector<BaseFighter*>& defenders) {
  int size = round(attackersSize * .01); // 5 defenders per 100 attackers
  for(int i = 0; i < size; i++) {
      BaseFighter* archer = new Archer();
      BaseFighter* warrior = new Warrior();
      BaseFighter* wizard = new Wizard();
      BaseFighter* elf = new Elf();
      BaseFighter* dragonborn = new DragonBorn();
      defenders.push_back(archer);
      defenders.push_back(warrior);
      defenders.push_back(wizard);
      defenders.push_back(elf);
      defenders.push_back(dragonborn);
    }
}

/*
*@brief: sets defender type to == attackers type
*@param: attacker*, defender*
*/
BaseFighter* getDefender(BaseFighter* attacker, vector<BaseFighter*>* defenders)
{
  for(BaseFighter* defender : *defenders) {
      if(defender->name == attacker->name) {
          return defender;
      }
  }
  return defenders->back();
}

/*
*@brief: calls heal function for current defending fighter
*/
void healDefenders(vector<BaseFighter*>& defenders) {
  for(BaseFighter* defender : defenders) {
      defender->heal();
  }
}
/*
*@brief: swaps current defender with another defender of same type if possible
*@param: defender*, defender* currently fighting
*@return: defender
*/
BaseFighter* swapDefender(vector<BaseFighter*>* defenders, BaseFighter*
curFighter) {
```

```
  for(BaseFighter* defender : *defenders) {
      if(defender != curFighter && defender->name == curFighter->name) {
          return defender;
      }
  }
  cout << "All " << curFighter->name << "'s are dead." << endl;
  return curFighter;
}

int main() {

  srand(time(0));                 // seed rand based on local time
  vector<BaseFighter*> attackers; // vector to be filled with attackers
  vector<BaseFighter*> defenders; // vector to be filled with defenders

  /*
  * fill vectors with fighters
  */
  populateAttackers(attackers);
  populateDefenders(attackers.size(), defenders);

  BaseFighter* CurrentAttacker;
  BaseFighter* CurrentDefender;

  while(attackers.size() && defenders.size()) {
    // get an attacker for this round
    CurrentAttacker = attackers.back();
    CurrentDefender = getDefender(CurrentAttacker, &defenders);
    while(CurrentAttacker->alive() && CurrentDefender->alive()) {

        // update game stats every slowly seconds to make it easy to follow
        this_thread::sleep_for(chrono::milliseconds(375));
        system("clear");
        cout << "***BolNiverse Fight Club***" << endl << endl;
        cout << "Attackers:" << setw(5) << " " << setw(0) << "Defenders:" << endl;
        cout << left << setw(13) << attackers.size() << " " << setw(0) <<
defenders.size() << endl;
        cout << endl << CurrentAttacker->name << ":" << CurrentAttacker->hp << "hp
| "
             << CurrentDefender->name << ":" <<CurrentDefender->hp << "hp" << endl
<< endl;

        // attackers start first
        int dmg = CurrentAttacker->attack();
        cout << "Attacking " << CurrentAttacker->name << " does: "
             << dmg << " DMG" << endl;

        CurrentDefender->damage(dmg);

        // if first defender is alive after first attacker dies, defender strikes
first
        if(CurrentDefender->alive()) {
            int dmg2 = CurrentDefender->attack();
            CurrentAttacker->damage(dmg2);
```

```
            cout << "Defending " << CurrentDefender->name << " survives and
counterattacks for: "
                 << dmg << " damage" << endl;
            cout << "Attacks health reduced to: " << CurrentAttacker->hp << endl;
            if(CurrentDefender->hp < 6) {
                // if the defender is low on health, swap him out for another
defender of the same type
                CurrentDefender = swapDefender(&defenders, CurrentDefender);
            }
        } else {
            // if the defender is dead, remove him from the defender list
            cout << "Defending " << CurrentDefender->name << " died" << endl;
            defenders.pop_back();
        }
        // heal the defenders
        healDefenders(defenders);
    }
    // deletes falled attacker from vector
    if(!CurrentAttacker->alive()) {
        attackers.pop_back();
        cout << "Attacking " << CurrentAttacker->name << " died" << endl;
    }

    int i = 0;
    for(BaseFighter* fighter : defenders) {
        if(fighter->hp <= 0) {
            defenders.erase(defenders.begin() + i);
            cout << "Defending " << CurrentDefender->name << " died" << endl;
        }
        i++;
    }
}
  // checks who won
  if(attackers.size()) {
      cout << endl << "Attackers Win" << endl;
  }
  if(defenders.size()) {
      cout << endl << "Defenders Win" << endl;
  }

}
```