Project 1: The MonsterList Data Structure

Purpose

The purpose of this project is to explore array structures and classes in C++. Additionally, this project will help you develop proficiency with searching and indexing in arrays.

Task

Download the starter code available online.

The starter code contains the following files:

Dice.hpp and **Monster.hpp**. These are the same classes we discussed in class. They have been separated into different header files.

MonsterList.hpp: This is the file where your code will go. There are 4 method stubs that you need to complete. This will be discussed below.

pokemon_small.tsv: This provides data for the program to use. DO NOT MODIFY this file. This file was modified from data provided the Github user, armgilles. The original is available here.

tester.hpp: The file runs tests on your MonsterList class. You should not modify this file other than to comment out specific tests while you test others.

Make sure that your code compiles correctly on your system before you start making modifications.

Your task is to …

1. Implement the **findMonsterIndexByName** method.
   ```
   int findMonsterIndexByName(std::string name)
   ```
   a. Given the monster's name, find the index of the monster.
   b. If the name is not part of our list, return -1.

2. Implement the **findMinAttackMonsterIndex** method.
   ```
   int findMinAttackMonsterIndex()
   ```
   a. Find the monster with the smallest attack power. This means the one with the smallest max attack dice.
   b. This will use Monster::getAttackMax for the Monster instances.
   c. Return the index of that Monster in the array.

3. Implement the **exchangeMonsters** method.
   ```
   void exchangeMonsters(int i, int j)
   ```
   a. Exchange the positions of the monsters in the array using indexes.
   b. Take the monster from position i, and save it to a temporary monster variable.
   c. Put the monster from position j into the i position in the array.
   d. Put the monster from the temporary variable into position j.

4. Implement the **findMinAttackMonsterInRange** method.
   ```
   int findMinAttackMonsterInRange(int start, int end)
   ```

   a. Find the monster with the smallest attack power in the given range.
   b. This will use Monster::getAttackMax for the Monster instances.
   c. Start checking for the minimum at the "start" index.
   d. Continue checking until you reach the "end" index.
   e. Make sure to actually check the monster in the "end" index or you may get an off-by-one error returning the wrong index.

Submission

Submit **<u>only</u>** your **MonsterList.hpp** file to Moodle.

Evaluation

You will be evaluated using the following criteria.

5 points	Correct **findMonsterIndexByName** method
5 points	Correct **findMinAttackMonsterIndex** method
5 points	Correct **exchangeMonsters** method
5 points	Correct **findMinAttackMonsterInRange** method

20 points total

Sample Output

When your MonsterList is implemented correctly, you should see the following output.

```
|----- TEST MonsterList::printAll() -----|
{Bulbasaur attack=4, Health=45}
{Ivysaur attack=5, Health=60}
{Venusaur attack=7, Health=80}
{Charmander attack=4, Health=39}
{Charmeleon attack=5, Health=58}
{Charizard attack=7, Health=78}
{Squirtle attack=4, Health=44}
{Wartortle attack=5, Health=59}
{Blastoise attack=7, Health=79}
{Caterpie attack=3, Health=45}
{Metapod attack=2, Health=50}
{Butterfree attack=4, Health=60}
{Weedle attack=3, Health=40}
{Kakuna attack=2, Health=45}
{Beedrill attack=8, Health=65}
{Pidgey attack=4, Health=40}
{Pidgeotto attack=5, Health=63}
{Pidgeot attack=7, Health=83}
{Rattata attack=5, Health=30}
{Raticate attack=7, Health=55}
{Spearow attack=5, Health=40}
{Fearow attack=8, Health=65}
```

```
{Ekans attack=5, Health=35}
{Arbok attack=7, Health=60}
{Pikachu attack=5, Health=35}
|-----  END MonsterList::printAll() -----|
|-----  ALREADY COMPLETED -----|

|----- TEST MonsterList::printMonsterAt() -----|
Should see: {Pikachu attack=5, Health=35}
Your code:
{Pikachu attack=5, Health=35}
|-----  END MonsterList::printMonsterAt() -----|
|-----  ALREADY COMPLETED -----|

|----- TEST MonsterList:: findMonsterIndexByName -----|
Searching for Beedrill
Should see: {Beedrill attack=8, Health=65}
Your code: {Beedrill attack=8, Health=65}

Searching for 'empty string'
Should see:
Your code:

|-----  END MonsterList:: findMonsterIndexByName -----|

|----- TEST MonsterList:: findMinAttackMonsterIndex -----|
Should see: {Metapod attack=2, Health=50}
Your code:
{Metapod attack=2, Health=50}
|-----  END MonsterList:: findMinAttackMonsterIndex -----|

|----- TEST MonsterList:: exchangeMonsters -----|
***** Before Exchange *****
3 : {Charmander attack=4, Health=39}
7 : {Wartortle attack=5, Health=59}
Exchange monsters 3 and 7 by calling  monsterList.exchangeMonsters(3, 7);
Should see:
3 : {Wartortle attack=5, Health=59}
7 : {Charmander attack=4, Health=39}
Your code:
3 : {Wartortle attack=5, Health=59}
7 : {Charmander attack=4, Health=39}
 ... Now exchanging back by calling monsterList.exchangeMonsters(3, 7); ...

|-----  END MonsterList:: exchangeMonsters -----|

|----- TEST MonsterList:: findMinAttackMonsterInRange -----|
Finding min attack monster between 1 and 6.
Should see: {Charmander attack=4, Health=39}
Your code:
{Charmander attack=4, Health=39}
Finding min attack monster between 11 and 13.
Should see: {Kakuna attack=2, Health=45}
Your code:
{Kakuna attack=2, Health=45}

|----- END MonsterList:: findMinAttackMonsterInRange -----|
```