Khadichabonu Valieva
Emily Robinson

Activity 5: Team Matchmaking

Classes.h

```
#ifndef C_PLUS_PLUS_CLASSES_H
#define C_PLUS_PLUS_CLASSES_H

#include <string>

using namespace std;

class Player {
public:
    Player() {
        this->name = "No name";
        this->skill = 0;
    };

    void SetName(string n) {
        this->name = n;
    }

    void SetSkill(int s) {
        this->skill = s;
    }

    string GetName() {
        return this->name;
    }

    int GetSkill() {
        return this->skill;
    }
```

```
private:
   string name;
   int skill;
};

#endif //C_PLUS_PLUS_CLASSES_H
```
Helpers.h
```
#ifndef C_PLUS_PLUS_HELPERS_H
#define C_PLUS_PLUS_HELPERS_H

#include <vector>
#include "classes.h"

using namespace std;

int GetLongestNameInTeam(vector<Player> team) {
   int maxLength = 0;
   for (int i = 0; i < team.size(); i++) {
      if (team[i].GetName().length() > maxLength) {
         maxLength = team[i].GetName().length();
      }
   }

   return maxLength;
}

void showTeams(vector<Player> team1, vector<Player> team2) {
   int LNT1 = GetLongestNameInTeam(team1);
   int LNT2 = GetLongestNameInTeam(team2);
   int offset = 10;
   int trailingOffset = 7;
   int leadingOffset = 5;

   int totalSkillT1 = 0;
   int totalSkillT2 = 0;

   cout << setw(leadingOffset + LNT1) << "Team 1";
```

```
    cout << setw(offset) << " ";
    cout << setw(trailingOffset + LNT2 - 2) << "Team 2";
    cout << endl;
    cout << setfill('-') << setw(leadingOffset + LNT1 + offset + LNT2 +
trailingOffset) << '-' << endl;

    for (int i = 0; i < team1.size(); i++) {
        cout << setfill(' ') << setw(leadingOffset + LNT1) <<
team1[i].GetName();
        cout << setw(offset) << " ";
        cout << setw(LNT2 + leadingOffset) << team2[i].GetName();
        cout << endl;

        totalSkillT1 += team1[i].GetSkill();
        totalSkillT2 += team2[i].GetSkill();
    }
    cout << setfill('-') << setw(leadingOffset + LNT1 + offset + LNT2 +
trailingOffset) << '-' << endl;

    cout << setfill(' ') << setw(leadingOffset + LNT1 - 2) << "T1 Skill: " <<
totalSkillT1;
    cout << setw(offset) << " ";
    cout << setw(LNT2 + leadingOffset - 2) << "T2 Skill: " << totalSkillT2;
}

vector<Player> SortQueue(vector<Player> queue) {
    vector<Player> result = vector<Player>();

    for (int i = 0; i < queue.size(); i++) {
        bool isInserted = false;
        for (int j = 0; j < result.size(); j++) {
            if (result[j].GetSkill() < queue[i].GetSkill()) {
                result.insert(result.begin() + j, queue[i]);
                isInserted = true;
                break;
            }
        }
```

```
        if (!isInserted) {
            result.push_back(queue[i]);
        }
    }

    return result;
}

#endif //C_PLUS_PLUS_HELPERS_H
```

Players.csv

```
slayer,4
deezfruits,6
sniper,9
megaman,10
bruiser,2
princess,8
skittles,4
thor,5
super,5
goku,8
```

Main.cpp

```
/*
1. Khadichabonu Valieva
2. Emily Robinson

Assignment: Activity 5
*/
```

```
#include <iostream>
#include <string>
#include <fstream>
#include <vector>
#include "classes.h"
#include "helpers.h"

using namespace std;

Player parsePlayerFromLine(string line) {
   Player player = Player();
   string name;
   string skill;
   bool nameFound = false;

   for (int i = 0; i < line.length(); i++) {
      if (line[i] == ',' || line[i] == ';') {
         nameFound = true;
         continue;
      }

      if (!nameFound) {
         name += line[i];
      } else {
         skill += line[i];
      }
   }

   player.SetName(name);
   player.SetSkill(stoi(skill));
   return player;
}

int main() {
   vector<Player> queue = vector<Player>();
   string filename = "../players.csv";
```

```
    string line;
    ifstream file;
    file.open(filename);

    if (file.is_open()) {
        while (getline(file, line)) {
            queue.push_back(parsePlayerFromLine(line));
        }
        file.close();
    } else {
        cout << "Unable to open file";
        return 0;
    }

    vector<Player> sortedQueue = SortQueue(queue);

    vector<Player> team1 = vector<Player>();
    vector<Player> team2 = vector<Player>();

    for (int i = 0; i < sortedQueue.size(); i++) {
        if (i % 2 == 0) {
            team1.push_back(sortedQueue[i]);
        }else{
            team2.push_back(sortedQueue[i]);
        }
    }

    showTeams(team1, team2);

    file.close();
    return 0;
}
```