# Team Roster Class with Operator Overloading

By

Jorge Jurado-Garcia

EE2510 Sec. 021, Spring 2021

Week 4&5 lab

Milwaukee School of Engineering

Submitted to:

Professor: Joshua D. Carl, Ph.D

EECS Department

Date Report Submitted: 04/11/21

# Objective

The objective of this lab is to create two classes Player and Team where the Team is a vector of the object Player named Roster with specific operators detailed by instructor. The Player class will also have operators when wanting to manipulate the players state. In both classes we have presented operator overloading.

# Description

In my I created extra functions in both objects to complete some operators. The Team Class has a default constructor, parameter constructor, copy constructor, and destructor. For my operator overloading for +=, -=, =, >, <, ==, <=, and >= . Where each operator has a set of manipulate for my private data Roster. Adding a Player in the end of the roster, subtracting a specific player, making a roster to another, less than, greater than, comparison, and less than and equal, and greater than and equal to. Likewise, for the Player class there operator overloading for arithmetic +, -,= and bool operators for >,<,<=,>=,==. Each adding up the stats of the Players health, power, speed, and intelligence and comparing the total. In my Player class I put a range for there stats where a player can have negative stats but can never be greater than 100.

# Conclusions

The lab was successful and was able to implement all the required classes and overloading their operator. The hardest part about this lab was understand how to use object overloading when using different objects and the use of vectors library in C++. The const Team& means I'm returning a reference const of my object Team. We added the const keyword so that the compiler will not modify the data and give us garbage data. Furthermore, I also found out a pointer to a variable being declared as a const can be assigned to a pointer that is declared as a const and this would be the same for functions as will. Once I was able to figure that out and understand that I can only use a function if that said function is declared to my class. For example, when trying to implement bool operators like this: bool Team:: operator<(const Team& tobj) const{ I never had a function that could have added all of the Players in Teams Roster. So when implemented I tried to read the Roster of object tobj and Team using a for loop by using a get_health(), get_etc(). This would not work because THOSE FUNCTIONS BELONG TO THE PLAYER CLASS. So, allow the index of that I am trying to read is as an object Player the Roster Vector itself is part of teams. So, I would get the error because the compile would see that the get_health() doesn't belong to teams so it would not exist. The way I went I went around this was by creating a int Team:: Total() for my team that reads the vector like a for-loop and implements a function called Total() from my Player class. This worked properly when having to compare player stats that were integers.

# Main code:

```
/*
Created: Jorge Jurado-Garcia
Date: 4/10/21
modifications:

*/

#include <iostream>
#include "Player.h"
#include "Team.h"
#include <vector>
using namespace std;

int main()
{
    Player P1("Bob",80,20,40,10);
    Player P2("Cat",10,10,10,0);
    Player P3("Rich",20,10,50,100);
    Player P4("Steve",0,1,0,1);
    Player Pc_add;
    Player Pc_sub;

    Pc_add = P1+P2;
    Pc_sub = P1-P2;
    cout<<"Pl stats:"<<P1.to_string()<<endl;
    cout<<"P2 stats:"<<P2.to_string()<<endl;
    cout<<"Pc+ stats:"<<Pc_add.to_string()<<endl;
    cout<<"Pc- stats"<<Pc_sub.to_string()<<endl;
    cout<<"P3 stats"<<P3.to_string()<<endl;
    cout<<"Making P3 equal to player Pc_sub"<<endl;
    P3 = Pc_sub;
    cout<<"P3 stats"<<P3.to_string()<<endl;
    cout<<"checking if P3 is equal to Pc_sub 1-true and 0-false"<<endl;
    bool x = P3==Pc_sub;
    bool y = P1==P2;
    cout<<x<<endl;
    cout<<"checing if P1 and P2 are equal 1-true 0-false"<<endl;
    cout<<y<<endl;
    cout<<"compare if P1>P2 1=true 0-false"<<endl;
    bool z = P1>P2;
    cout<<z<<endl;
    cout<<"compare if P1<P2 1=true 0-false"<<endl;
    bool zz = P1<P2;
    cout<<zz<<endl;
    cout<<"comparing if P3<=Pc_sub 1-true 0-false"<<endl;
    bool zzz = P3<=Pc_sub;
    cout<<zzz<<endl;
    cout<<"this would the same for P3>=Pc_sub since they are the same players
essentially"<<endl;
    bool zzzz = P3>=Pc_sub;
    cout<<zzzz<<endl;

    vector<Player> Roster1{P1,P2,P3};
    Team T1(Roster1);
```

```
T1.to_string();
cout<<"Adding a Player"<<endl;
T1+=P4;
T1.to_string();

cout<<"subtracting a Player"<<endl;
T1-=P1;
T1-=P2;

cout<<"Team 1 are:"<<endl;
T1.to_string();
int y2 = T1.total();
cout<<y2<<endl;

cout<<"Team 2 are:"<<endl;
vector<Player> Roster2{P1,P2};
Team T2(Roster2 );
T2.to_string();
int y3 = T2.total();
cout<<y3<<endl;

bool out;
out = T2<T1;
cout<<"Team2<Team1 1-true 0-false"<<endl<<out<<endl;

 bool out2;
out2 = T2>T1;
cout<<"Team2>Team1 1-true 0-false"<<endl<<out2<<endl;

bool out4;
out4 = T2==T1;
cout<<"is T2==T1 1-true 0-false"<<endl<<out4<<endl;

cout<<"clearing Team2"<<endl;
T2.clearPlayers();
T2.to_string();

 cout<<"Team2 = Team1 is"<<endl;
T2 = T1;
T2.to_string();
bool out5;
out5 = T1==T2;
cout<<"is T2==T1? 1-true 0-false"<<endl<<out5<<endl;

cout<<"comparing teams"<<endl;
Player P20("Jorge",1,1,1,1);
Player P30("Carl",100,100,100,100);
Player P50("Pancho",10,23,10,0);
Player P60("Crazy_Carl",100,100,100,100);
Player P70("Carl",1,1,1,1);
Player P80("Carl",2,2,2,2);

vector<Player> R1{P20,P30,P70,P50,P60};
vector<Player> R2{P20,P30,P70,P50,P60};
Team T3(R1);
Team T4(R2);
cout<<"Team 3 size||"<<endl;
```

```
    T3.to_string();
    cout<<"Team 4 size||"<<endl;
    T4.to_string();

    bool out7;
    out7 = T3 >= T4;
    cout<<"when T3 >= T4 the answer is 1-true 0-false: "<<out7<<endl;

    bool out8;
    out8 = T3 <= T4;
    cout<<"when T3 <= T4 the answer is 1-true 0-false: "<<out8<<endl;
    cout<<"taking out carl(1,1,1,1) and replacing carl(2,2,2,2)"<<endl;
    T4 -=P70;

    T4 += P80;
    T4.to_string();

    bool out9;
    out9 = T3<=T4;
    cout<<"when T3 <= T4 the answer is 1-true 0-false: "<<out9<<endl;

    bool out6;
    out6 = T3 >= T4;
    cout<<"when T3 >= T4 the answer is 1-true 0-false: "<<out6<<endl;
    return 0;
}
```

# Player Class Definitions

```
/*
Created: Jorge Jurado-Garcia
Date: 4/3/21
modifications:
4/11/21 adding name string as a private variable and function called total.
*/
#ifndef PLAYER_H_INCLUDED
#define PLAYER_H_INCLUDED
#include <stdio.h>
#include <string>
#include <iostream>

using namespace std;

class Player{

private:
    int health;
    int power;
    int speed;
    int intelligence;
    string name;

public:
    Player(); //generic constructor
    Player(string n,int h, int p, int s, int intel); //parameter constructor
    Player(const Player &PlayerCopy); //copy constructor
    virtual ~Player(); //destructor

    int total() const;
    int get_health();
    int get_power();
    int get_speed();
    int get_intelligence();
    string get_name() const;

    set_health(int h);
    set_power(int p);
    set_speed(int s);
    set_intelligence(int intel);
    set_name(string n);

    const Player operator+(const Player& obj) const;
    const Player operator-(const Player& obj) const;
    const Player& operator=(const Player& obj);
    bool operator==(const Player& obj) const;
    bool operator<(const Player& obj) const;
    bool operator>(const Player& obj) const;
    bool operator<=(const Player& obj) const;
    bool operator>=(const Player& obj) const;

    string to_string() const;
};
#endif // PLAYER_H_INCLUDED
```

# Player Class Methods

```
/*
 * Player_functions.ccp
 *
 *  Created on: April 3, 2021
 *      Author: Jorge Jurado-Garcia
 *
 * Defines the functions for player.h hile
 *
 */

#include "Player.h"
#include <stdio.h>
#include <string>
#include <math.h>
#include <iostream>
#include <sstream>

using namespace std;

Player::Player(){
   // cout<<"***In default constructor****"<<endl;
   health = 0;
   power = 0;
   speed= 0;
   intelligence = 0;
   name = "no name";
}
Player::Player(string n,int h, int p, int s, int intel){
   //cout<<"***In parameter constructor***"<<endl;
   set_health(h);
   set_power(p);
   set_speed(s);
   set_intelligence(intel);
   set_name(n);
}
Player::Player(const Player &PlayerCopy){
  // cout<<"**copy constructor***"<<endl;
    this->health = PlayerCopy.health;
    this->power = PlayerCopy.power;
    this->speed = PlayerCopy.speed;
    this ->intelligence = PlayerCopy.intelligence;
    this ->name = PlayerCopy.name;
}
Player:: ~Player(){
}
int Player:: get_health(){
   return health;
}
int Player:: get_power(){
    return power;
}
int Player:: get_speed(){
    return speed;
```

```
}
int Player:: get_intelligence(){
    return intelligence;
}
string Player:: get_name()const{
    return name;
}
Player::  set_health(int h){
    health = h;
    return 0;
}
Player:: set_power(int p){
    power = p;
    return 0;
}
Player:: set_speed(int s){
    speed = s;
    return 0;
}
Player:: set_intelligence(int intel){
    intelligence = intel;
    return 0;
}
Player:: set_name(string n){
    name = n;
    return 0;
}
const Player Player::operator+(const Player& obj) const{
    string super_name = this->name+"+"+obj.name;
    int super_health = this->health + obj.health;
    int super_power = this->power + obj.power;
    int super_speed = this->speed + obj.speed;
    int super_intelligence = this->intelligence + obj.intelligence;
    if( ( (super_health) || (super_power) || (super_speed) ||
(super_intelligence) ) > 100){
        super_health = 100;
        super_power = 100;
        super_speed = 100;
        super_intelligence = 100;
    }
    Player S1(super_name,super_health, super_power, super_speed,
super_intelligence);
    return S1;
}
const Player Player::operator-(const Player& obj) const{
     string super_name = this->name+"-"+obj.name;
    int super_health, super_power, super_speed, super_intelligence;
    super_health = this->health - obj.health;
    super_power = this->power - obj.power;
    super_speed = this->speed - obj.speed;
    super_intelligence = this->intelligence - obj.intelligence;
    Player S1(super_name,super_health, super_power, super_speed,
super_intelligence);
    return S1;
}
const Player& Player:: operator=(const Player& obj){
    if(this != &obj){
```

```
        health = obj.health;
        power = obj.power;
        speed = obj.speed;
        intelligence = obj.intelligence;
        name = obj.name;
    }
    return *this;
}
bool Player:: operator==(const Player& obj) const{
    //If both the operands are non-zero, then the condition becomes true.
    //compates if they are equal to each other
    //if false the return value will be zero else one
    bool health_part = this->health == obj.health;
    bool power_part = this->power == obj.power;
    bool speed_part = this->speed == obj.speed;
    bool intel_part = this->intelligence == obj.intelligence;
    bool name_part = this->name == obj.name;
    //checks for all of the values return true
    //if all the same false if different
    return( name_part && health_part && power_part && speed_part &&
intel_part);
}
bool Player:: operator<(const Player& obj) const{
    float total1 = (this->total());
    float total2 = (obj.total());
    bool outcome = total1 < total2;
    return outcome;
}
bool Player:: operator>(const Player& obj) const{
    float total1 = this->total();
    float total2 = obj.total();
    bool outcome = total1 > total2;
    return outcome;
}
bool Player:: operator<=(const Player& obj) const{
    float total1 = this->total();
    float total2 = obj.total();
    bool outcome;
    if( (total1 == total2) || (total1 < total2) ){
        outcome = true;
    }
    else{
        outcome = false;
    }
    return outcome;
}
bool Player:: operator>=(const Player& obj) const{
    float total1 = this->total();
    float total2 = total();
    bool outcome;
    if( (total1 == total2) || (total1 > total2) ){
        outcome = true;
    }
    else{
        outcome = false;
    }
    return outcome;
```

```
}
string Player:: to_string() const{
    ostringstream convert_h;
    ostringstream convert_p;
    ostringstream convert_s;
    ostringstream convert_i;
    convert_h << health;
    convert_p << power;
    convert_s << speed;
    convert_i << intelligence;
    string output = name+"||"+"Health:" + convert_h.str() + " Power:" +
convert_p.str();
    output = output + " Speed:" + convert_s.str() + " Intel:" +
convert_i.str();
    return output;
}
int Player:: total() const{
    int totals = this->health+this->speed+this->power+this->intelligence;
    return totals;
}
```

# Player Class Definitions

```
/*
Created: Jorge Jurado-Garcia
Date: 4/3/21
modifications:
4/8/21 create code for operator of += -= =
4/11/21 creating the functions needed for bool operators
*/

#ifndef TEAM_H_INCLUDED
#define TEAM_H_INCLUDED

#include <stdio.h>
#include <string>
#include <vector>
#include <iostream>
#include "Player.h"

using namespace std;

class Team{
private:
    vector<Player> Roster;

public:
    string get_name() const;
    int total()const;

    Team(); //default constructor
    Team(vector<Player> NewRoster); //creating a new roster
    Team(Team &otherTeamCopy); //copy constructor)
    virtual ~Team(); //destructor

    //will need set and get functions for roaster;
    set_Roster(vector<Player> R);
    vector<Player> get_Roster();

    const Team& operator+=(const Player& Ply);
    const Team& operator-=(const Player& Ply);
    const Team& operator=(const Team& tobj);

    bool operator==(const Team& tobj) const;
    bool operator<(const Team& tobj) const;
    bool operator>(const Team& tobj) const;
    bool operator<=(const Team& tobj) const;
    bool operator>=(const Team& tobj) const;

    clearPlayers();
    to_string() const;
};
#endif // TEAM_H_INCLUDED
```

# Player Class Methods

```
/*
 * Team_functions.ccp
 *  Created on: April 3, 2021
 *      Author: Jorge Jurado-Garcia
 * Defines the functions for Team.h hile
 */

 #include "Team.h"
 #include "Player.h"
 #include <stdio.h>
 #include <string>
 #include <vector>
 #include <math.h>
 #include <iostream>
 #include <sstream>

 using namespace std;
string Team:: get_name() const{
string Name;
    int sizel = Roster.size();
    for(int i=0; i< sizel; ++i){
            Name = Name+Roster[i].get_name();
    }
    return Name;
}
int Team:: total()const{
    int total = 0;
    int sizel = Roster.size();
    for(int i=0; i< sizel; ++i){
            total = Roster[i].total()+total;
    }
    return total;
}
 Team::Team(){
     cout<<"***In default constructor****"<<endl;
     //need to create a roster of zero players
     vector<Player> Roster(0);

 }
 Team::Team(vector<Player> NewRoster){
   //here we are given a vector of player from main
   //creating a team for them
   cout<<"***In parameter constructor***"<<endl;
    set_Roster( NewRoster );
 }
Team::Team(Team &otherTeamCopy){
    //going to have to have a Team called otherTeamcopy and that will have
    //a vector full of players
    cout<<"**copy constructor***"<<endl;
   this->Roster = otherTeamCopy.Roster;
}
 Team:: set_Roster(vector<Player> R){
    //sizing the roster vector to the size of R vector
    Roster.resize(R.size());
```

```
    for(unsigned int i=0; i < R.size(); i++){
        //items in vec2 will be in roster copy
        Roster[i] = R[i];
    }
return 0;
}
vector<Player> Team::get_Roster(){
    //return the vector of
    return Roster;
}
 Team:: ~Team(){
 }
 const Team& Team::operator+=(const Player& Ply){
    this->Roster.push_back(Ply);
    return *this;
 }
const Team& Team:: operator-=(const Player& Ply){
    for(auto i = Roster.begin(); i != Roster.end(); ++i){
        if(*i == Ply){
        this->Roster.erase(i);
        }
    }
cout<<"Size of Vector after removal:"<<Roster.size()<<endl;
return *this;
}
const Team& Team:: operator=(const Team& tobj){
    if(this != &tobj){
        Roster = tobj.Roster;
    }
    return *this;
}
Team:: clearPlayers(){
    Roster.clear();
    return 0;
}
Team:: to_string() const{
    vector<Player>::const_iterator iter;
    for(iter = Roster.begin(); iter != Roster.end(); iter++){
        cout<<"\t"<<(*iter).to_string()<<endl;
    }
    return 0;
}
bool Team:: operator<(const Team& tobj) const{
    int total2 = tobj.total();
    int total1 = this->total();
    if(total1 < total2){
        return true;
    }
    else{
        return false;
    }
}
bool Team:: operator>(const Team& tobj) const{
    int total2 = tobj.total();
    int total1 = this->total();
    if(total1 > total2){
        return true;
```

```
    }
    else{
        return false;
    }
}
bool Team:: operator==(const Team& tobj) const{
    bool out;
    int size1 = this->Roster.size();
    int size2 = tobj.Roster.size();
    if(size1 == size2){
        for(int i=0; i< size1; ++i){
            if(this->Roster[i]== tobj.Roster[i]){
                //if the player are the same then do nothing;
                out = true;
            }
            else{
                return false;
            }
        }
        return out;
    }
    else{
        return false;
    }
}
bool Team:: operator<=(const Team& tobj) const{
    string name1 = this->get_name();
    string name2 = tobj.get_name();
    int total1 = this->total();
    int total2 = tobj.total();
    if( ( (total1 < total2) || (total1==total2) ) &&
(name1.compare(name2)==0) ){
        return true;
    }
    else if((total1 < total2) || (total1==total2) ){
        return true;
    }
    else{
        return false;
    }
}
bool Team:: operator>=(const Team& tobj) const{
    string name1 = this->get_name();
    string name2 = tobj.get_name();
    int total1 = this->total();
    int total2 = tobj.total();
    if( ( (total1 > total2) || (total1==total2) ) &&
(name1.compare(name2)==0) ){
        return true;
    }
    else if((total1 > total2) || (total1==total2) ){
        return true;
    }
    else{
        return false;
    }
}
```

Console Result:

Pl stats:Bob||Health:80 Power:20 Speed:40 Intel:10

P2 stats:Cat||Health:10 Power:10 Speed:10 Intel:0

Pc+ stats:Bob+Cat||Health:90 Power:30 Speed:50 Intel:10

Pc- statsBob-Cat||Health:70 Power:10 Speed:30 Intel:10

P3 statsRich||Health:20 Power:10 Speed:50 Intel:100

Making P3 equal to player Pc_sub

P3 statsBob-Cat||Health:70 Power:10 Speed:30 Intel:10

checking if P3 is equal to Pc_sub 1-true and 0-false

1

checing if P1 and P2 are equal 1-true 0-false

0

compare if P1>P2 1=true 0-false

1

compare if P1<P2 1=true 0-false

0

comparing if P3<=Pc_sub 1-true 0-false

1

this would the same for P3>=Pc_sub since they are the same players essentially

1

***In parameter constructor***

Bob||Health:80 Power:20 Speed:40 Intel:10

Cat||Health:10 Power:10 Speed:10 Intel:0

Bob-Cat||Health:70 Power:10 Speed:30 Intel:10

Adding a Player

Bob||Health:80 Power:20 Speed:40 Intel:10

Cat||Health:10 Power:10 Speed:10 Intel:0

```
	Bob-Cat||Health:70 Power:10 Speed:30 Intel:10
	Steve||Health:0 Power:1 Speed:0 Intel:1
subtracting a Player
Size of Vector after removal:3
Size of Vector after removal:2
Team 1 are:
	Bob-Cat||Health:70 Power:10 Speed:30 Intel:10
	Steve||Health:0 Power:1 Speed:0 Intel:1
122
Team 2 are:
***In parameter constructor***
	Bob||Health:80 Power:20 Speed:40 Intel:10
	Cat||Health:10 Power:10 Speed:10 Intel:0
180
Team2<Team1 1-true 0-false
0
Team2>Team1 1-true 0-false
1
is T2==T1 1-true 0-false
0
clearing Team2
Team2 = Team1 is
	Bob-Cat||Health:70 Power:10 Speed:30 Intel:10
	Steve||Health:0 Power:1 Speed:0 Intel:1
is T2==T1? 1-true 0-false
1
comparing teams
***In parameter constructor***
```

```
***In parameter constructor***
Team 3 size||
    Jorge||Health:1 Power:1 Speed:1 Intel:1
    Carl||Health:100 Power:100 Speed:100 Intel:100
    Carl||Health:1 Power:1 Speed:1 Intel:1
    Pancho||Health:10 Power:23 Speed:10 Intel:0
    Crazy_Carl||Health:100 Power:100 Speed:100 Intel:100
Team 4 size||
    Jorge||Health:1 Power:1 Speed:1 Intel:1
    Carl||Health:100 Power:100 Speed:100 Intel:100
    Carl||Health:1 Power:1 Speed:1 Intel:1
    Pancho||Health:10 Power:23 Speed:10 Intel:0
    Crazy_Carl||Health:100 Power:100 Speed:100 Intel:100
when T3 >= T4 the answer is 1-true 0-false: 1
when T3 <= T4 the answer is 1-true 0-false: 1
taking out carl(1,1,1,1) and replacing carl(2,2,2,2)
Size of Vector after removal:4
    Jorge||Health:1 Power:1 Speed:1 Intel:1
    Carl||Health:100 Power:100 Speed:100 Intel:100
    Pancho||Health:10 Power:23 Speed:10 Intel:0
    Crazy_Carl||Health:100 Power:100 Speed:100 Intel:100
    Carl||Health:2 Power:2 Speed:2 Intel:2
when T3 <= T4 the answer is 1-true 0-false: 1
when T3 >= T4 the answer is 1-true 0-false: 0

Process returned 0 (0x0)   execution time : 0.089 s
Press any key to continue.
```

"C:\Users\jurado-garciaj\Documents\MSOE COURSES\EE COURSES\EE2510\projects\CODE\LAB1\LAB4_RERUN_EE2520\bin\Debug\LAB4_RERUN_EE2520.exe"

```
Pl stats:Bob||Health:80 Power:20 Speed:40 Intel:10
P2 stats:Cat||Health:10 Power:10 Speed:10 Intel:0
Pc+ stats:Bob+Cat||Health:90 Power:30 Speed:50 Intel:10
Pc- statsBob-Cat||Health:70 Power:10 Speed:30 Intel:10
P3 statsRich||Health:20 Power:10 Speed:50 Intel:100
Making P3 equal to player Pc_sub
P3 statsBob-Cat||Health:70 Power:10 Speed:30 Intel:10
checking if P3 is equal to Pc_sub 1-true and 0-false
1
checing if P1 and P2 are equal 1-true 0-false
0
compare if P1>P2 1=true 0-false
1
compare if P1<P2 1=true 0-false
0
comparing if P3<=Pc_sub 1-true 0-false
1
this would the same for P3>=Pc_sub since they are the same players essentially
1
***In parameter constructor***
        Bob||Health:80 Power:20 Speed:40 Intel:10
        Cat||Health:10 Power:10 Speed:10 Intel:0
        Bob-Cat||Health:70 Power:10 Speed:30 Intel:10
Adding a Player
        Bob||Health:80 Power:20 Speed:40 Intel:10
        Cat||Health:10 Power:10 Speed:10 Intel:0
        Bob-Cat||Health:70 Power:10 Speed:30 Intel:10
        Steve||Health:0 Power:1 Speed:0 Intel:1
subtracting a Player
Size of Vector after removal:3
Size of Vector after removal:2
Team 1 are:
        Bob-Cat||Health:70 Power:10 Speed:30 Intel:10
        Steve||Health:0 Power:1 Speed:0 Intel:1
122
Team 2 are:
***In parameter constructor***
        Bob||Health:80 Power:20 Speed:40 Intel:10
        Cat||Health:10 Power:10 Speed:10 Intel:0
180
Team2<Team1 1-true 0-false
0
Team2>Team1 1-true 0-false
1
is T2==T1 1-true 0-false
0
clearing Team2
Team2 = Team1 is
        Bob-Cat||Health:70 Power:10 Speed:30 Intel:10
        Steve||Health:0 Power:1 Speed:0 Intel:1
is T2==T1? 1-true 0-false
1
comparing teams
***In parameter constructor***
***In parameter constructor***
Team 3 size||
        Jorge||Health:1 Power:1 Speed:1 Intel:1
        Carl||Health:100 Power:100 Speed:100 Intel:100
        Carl||Health:1 Power:1 Speed:1 Intel:1
        Pancho||Health:10 Power:23 Speed:10 Intel:0
        Crazy_Carl||Health:100 Power:100 Speed:100 Intel:100
Team 4 size||
        Jorge||Health:1 Power:1 Speed:1 Intel:1
```