Shallow Copy vs Deep Copy

A shallow copy simply performs a straight element by element copy of all data in the object. The memory addresses of pointers will also be copied so the duplicate object will contain pointers to the same memory.

A deep copy also duplicates the data on the heap that the pointers point to (the pointee).

1. Using the following class, write two functions which copy the Player class, one that performs a shallow copy and one that performs a deep copy.

```
#include <iostream>

#define NAME_LENGTH 32
class Weapon
{
public:
	int m_Strength;
	float m_Speed;
};

class Player
{
public:
	Player* GetShallowCopy()
	{
		//Create and return a shallow copy here
	}

	Player* GetDeepCopy()
	{
		//Create and return a deep copy here.
	}

	char* m_Name;
	Weapon* m_Weapon;
	int m_Health;
};
```

2. Now test that your copies are working using this code:
   After changing a character in player1's name, the name for player2 (shallow copy) will also be changed because they share the same memory for their name. player3 (deep copy) will remain unchanged, it has its own memory that is not connected to player 1 in any way.

```
void main()
{
	//Create first player
	Player* player1 = new Player();

	//Assign name
	player1->m_Name = new char[NAME_LENGTH];
	strcpy_s(player1->m_Name, NAME_LENGTH, "John");

	//Create weapon
	player1->m_Weapon = new Weapon();
	player1->m_Weapon->m_Strength = 3;
	player1->m_Weapon->m_Speed = 1.5f;

	//Initialise health
	player1->m_Health = 100;

	//Create second player (Shallow copy)
	Player* player2 = player1->GetShallowCopy();

	//Create third player (Deep copy)
	Player* player3 = player1->GetDeepCopy();

	std::cout << "Initial Values" << std::endl;
	std::cout << player1->m_Name << std::endl;                    //"John"
	std::cout << player2->m_Name << std::endl;                    //"John"
	std::cout << player3->m_Name << std::endl << std::endl;       //"John"

	player1->m_Name[1] = ' '; //Replacing the 'o' character with a space.

	std::cout << "New Values" << std::endl;
	std::cout << player1->m_Name << std::endl;                    //"J hn"
	std::cout << player2->m_Name << std::endl;                    //"J hn"
	std::cout << player3->m_Name << std::endl;                    //"John"

	system("pause");
}
```

3. Write code to test that the same works for the Weapon as well. Note that the Health won't behave the same because it doesn't involve a pointer: changing player1's health won't change player2's. Primitive types are effectively always deep copied and the difference between shallow and deep only applies to pointers.