Lee Zong Yang 100073484

Table of Contents

1. Task 1
2. Task 2
3. Task 3
4. Task 4
5. Task 5

**Task 1**

**Description**

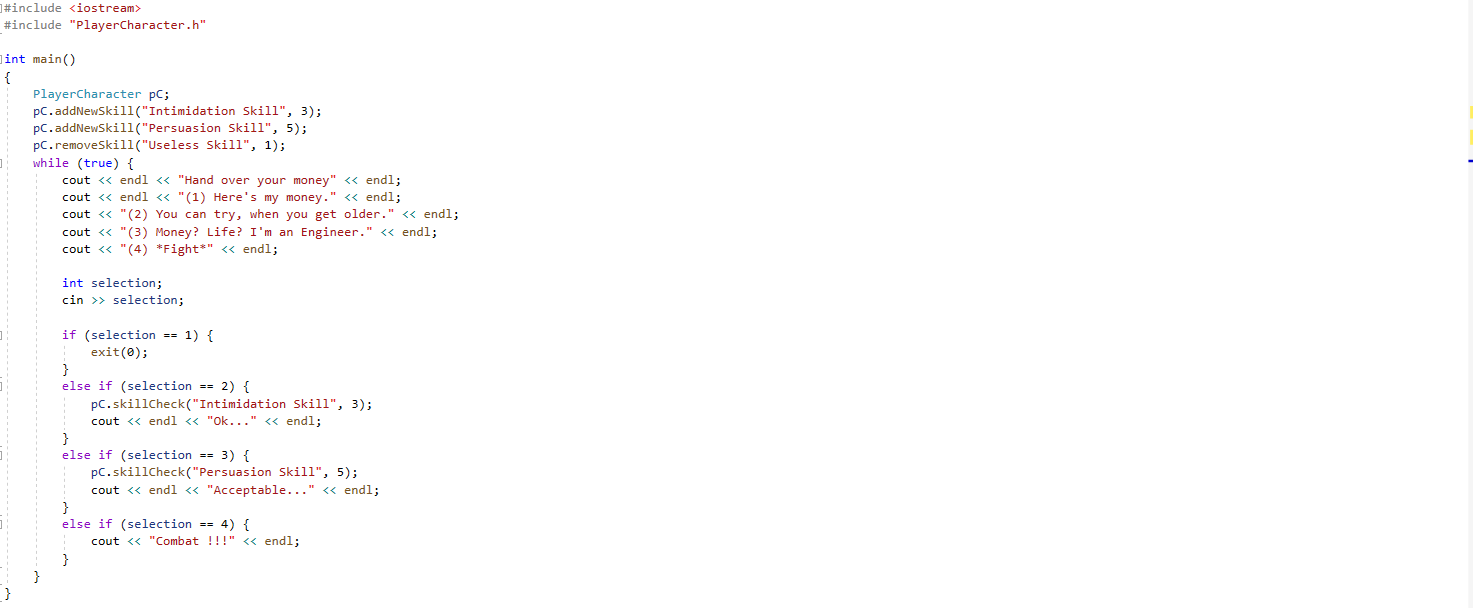
Implement an event loop. Define a class of player’s character and instantiate an object to represent the player during the event loop. The class should have stub methods that can check whether a skill is existed, add and remove skill. Testing should be done too.

**Concept**

I choose while loop instead of GOTO because it is a better programming practice. Error handling of input had been implemented too. At this step, only stub methods are required so I just write some dummy code inside the stub methods.

**Implementation**





**Output**

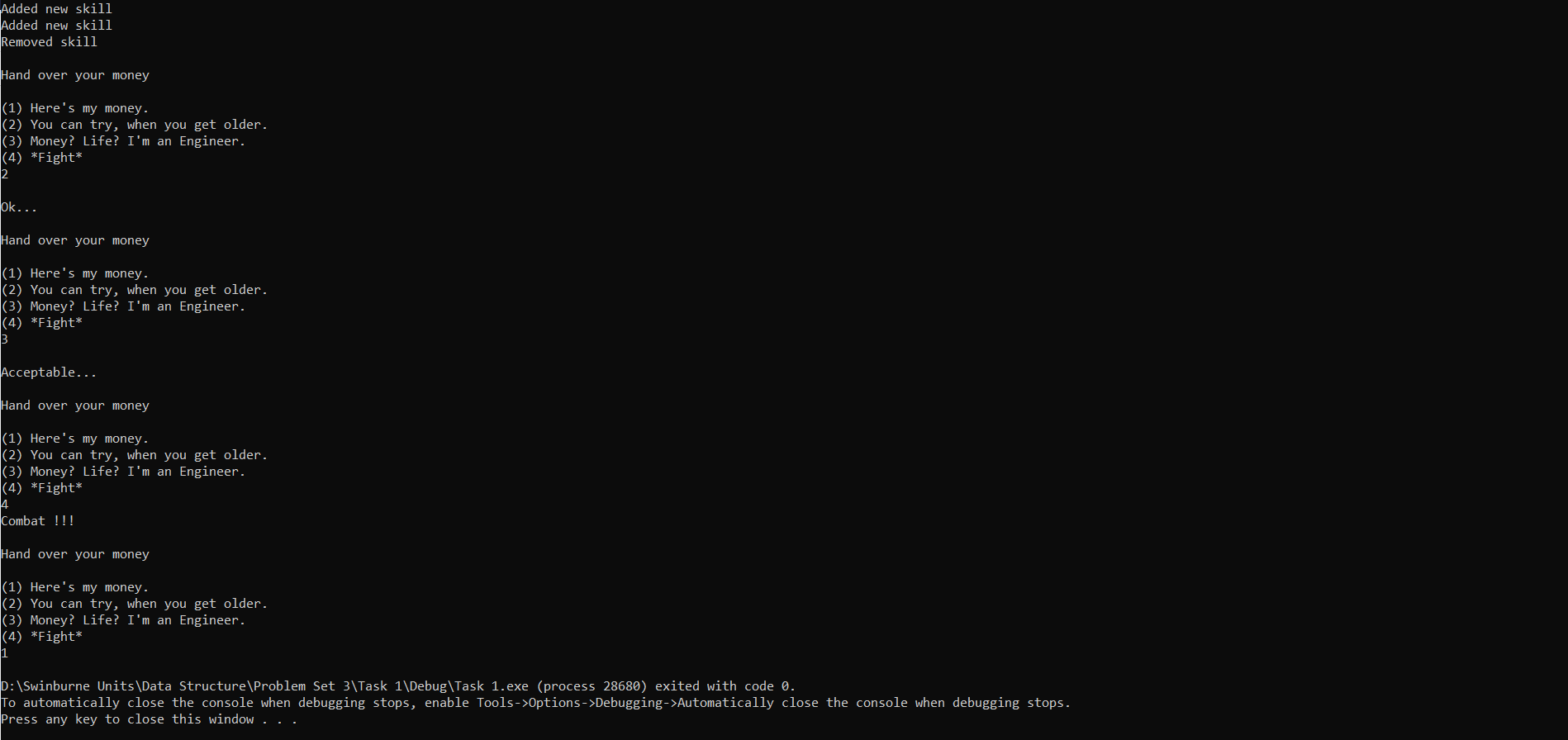


Figure: console output

**Troubleshooting**

Like the task I attempted before so no problem.

**Task 2**

**Description**

The following tasks do require a singly linked-list or doubly linked-list, so a class to represent a node in list need to complete in this task. The node should fulfil the requirement of holding the data and pointer to next node.

**Concept**

The reason I choose Singly Linked-List over Doubly Linked-List is zero need to traverse the list in two direction. With a doubly linked-list, it would take more space for each node as an extra pointer to previous node is required. The main operation is to find whether a skill exist or not, so it is O(n) time complexity for both singly and doubly.

**Implementation**

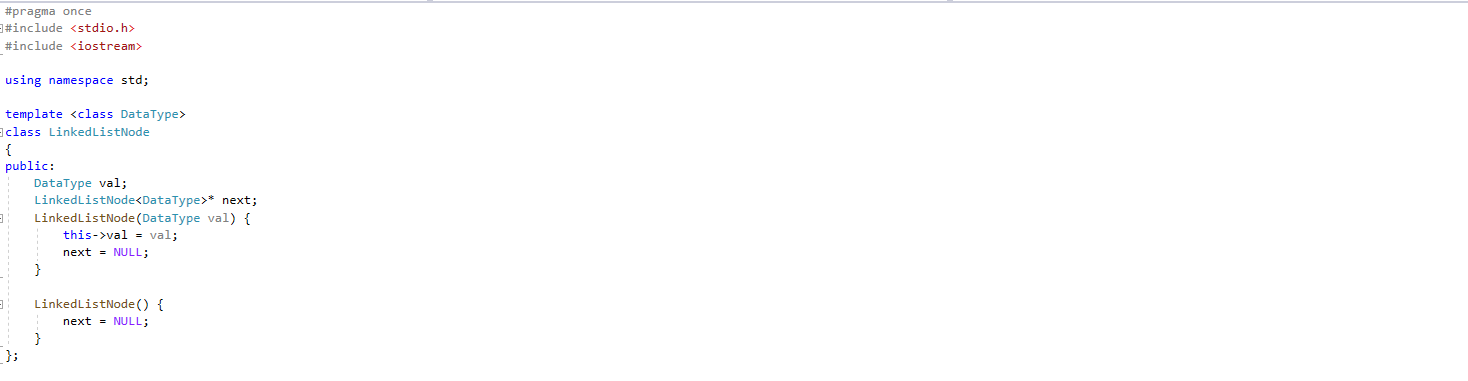


Figure: Linked list node (LinkedListNode.h)



Figure: main function (main.cpp)

**Output**

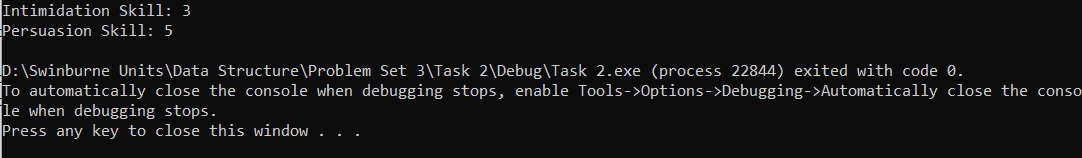


Figure: console output

**Troubleshooting**

To make the node as generic as possible, I found out template class is the way to go.

**Task 3**

**Description**

Implement a linked list, either directly into the PlayerCharacter class or an iterator to manage the linked list. Adequate interface and functionalities should be implemented for the stub functions of PlayerCharacter, namely Skill Check, Skill Add, and Skill Remove. Testing should be done.

**Concept**

Singly linked list is used to solve this problem. In my implementation, there is head and tail pointer, thus the time complexity to prepend and append is both O(1). The linked list consists solely of data and pointer to the next node. Therefore, only added skill will increase the storage cost of linked list and potential unused space problem in array will not happen. To prepend and append in array, the time complexity is O(n) as it is needed to move all elements in the array to arrange for the position. There is a need to search for certain skill, and the time complexity for the operation is O(n) for linked list, but it is O(n) for array too, as the position of the item is not known and traverse through the whole array is needed in the worst-case scenario. Frequent delete at certain index operation is carried out when skill is removed, and linked-list triumph over array with O(1) time complexity.

I did not choose to implement an iterator to manage the linked list is due to the idiom “Pre-optimization is the root of devil”. Iterator could enhance the open closed principle, but there is no need at this moment as the purpose of this program is clear, to finish problem set 3.

**Implementation**

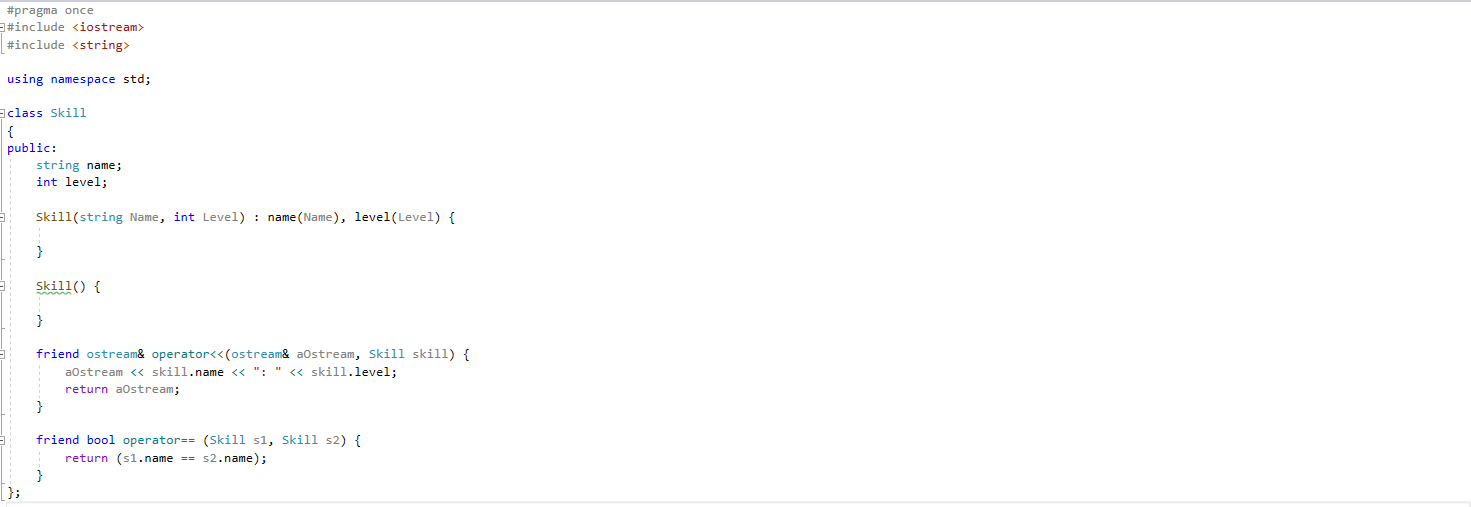


Figure: Skill class (Skill.h)

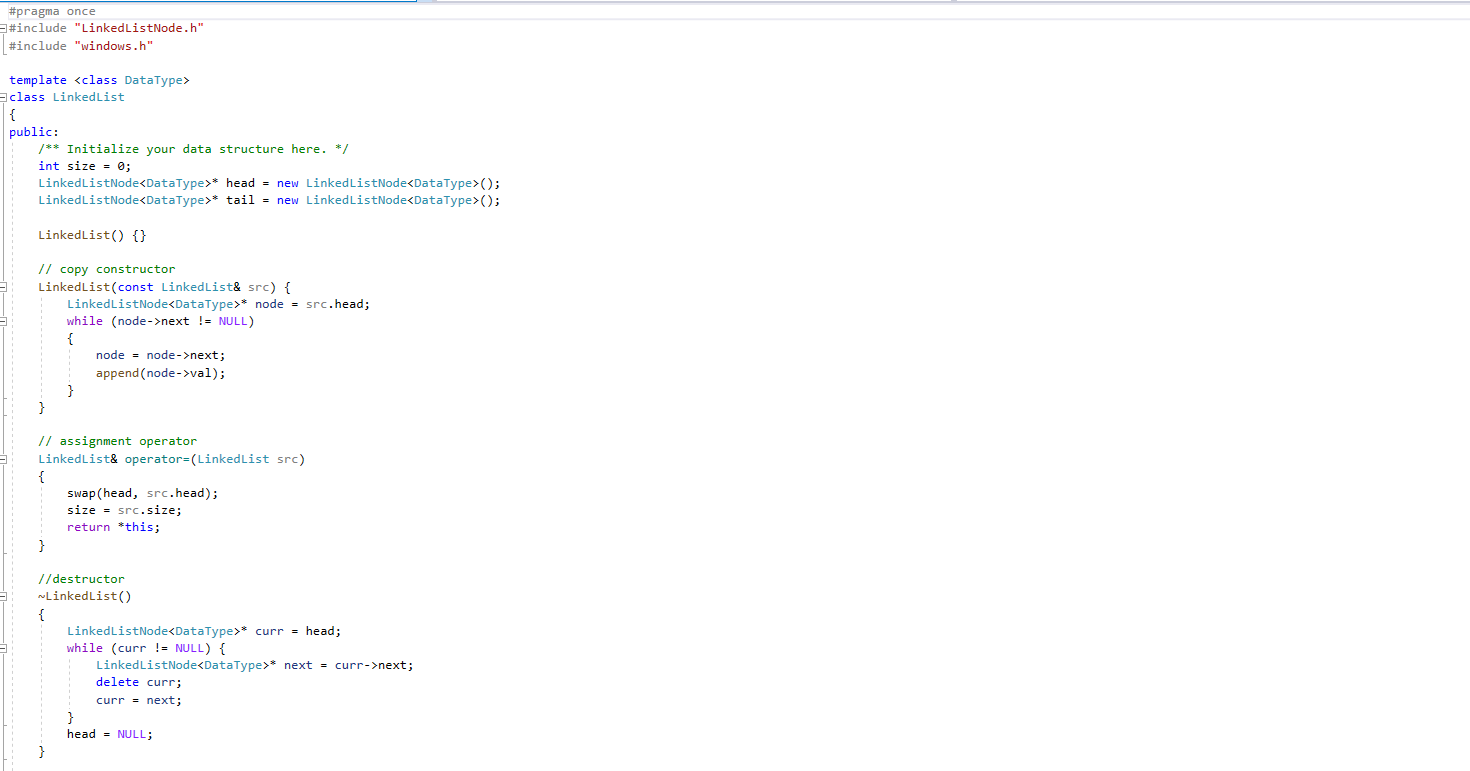
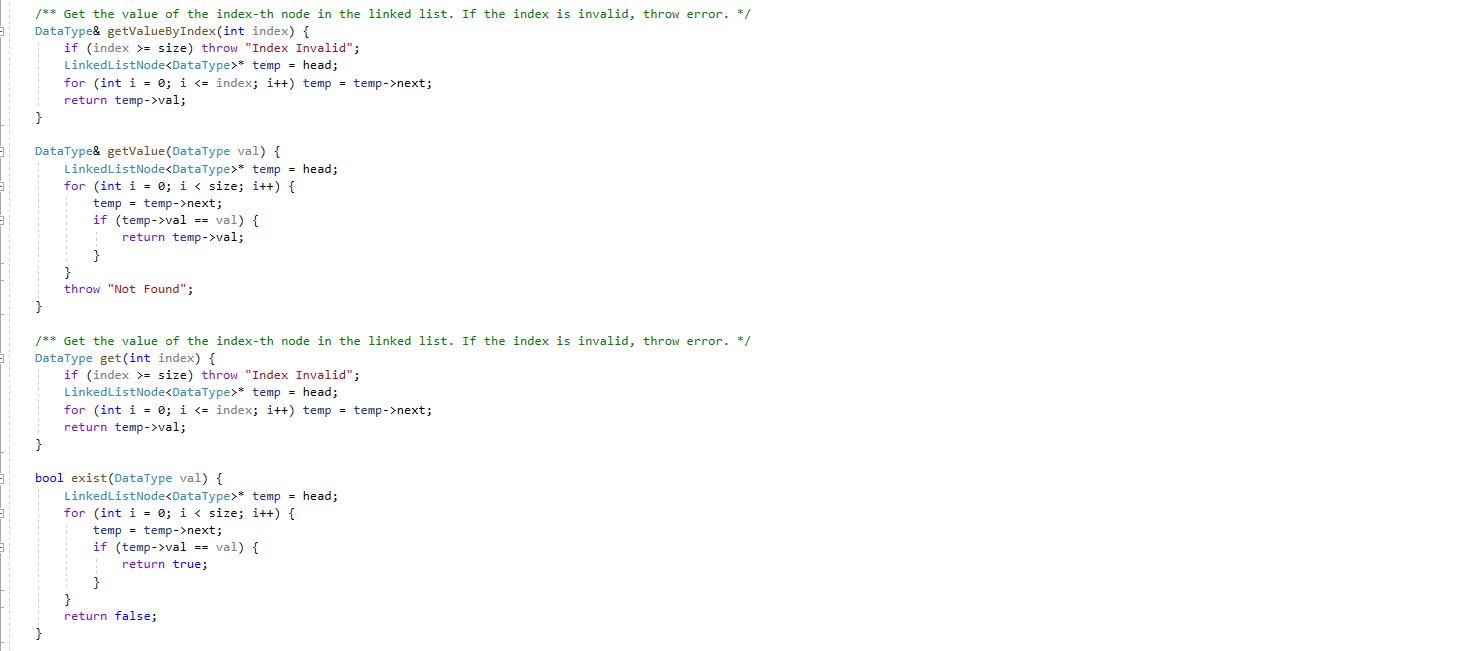
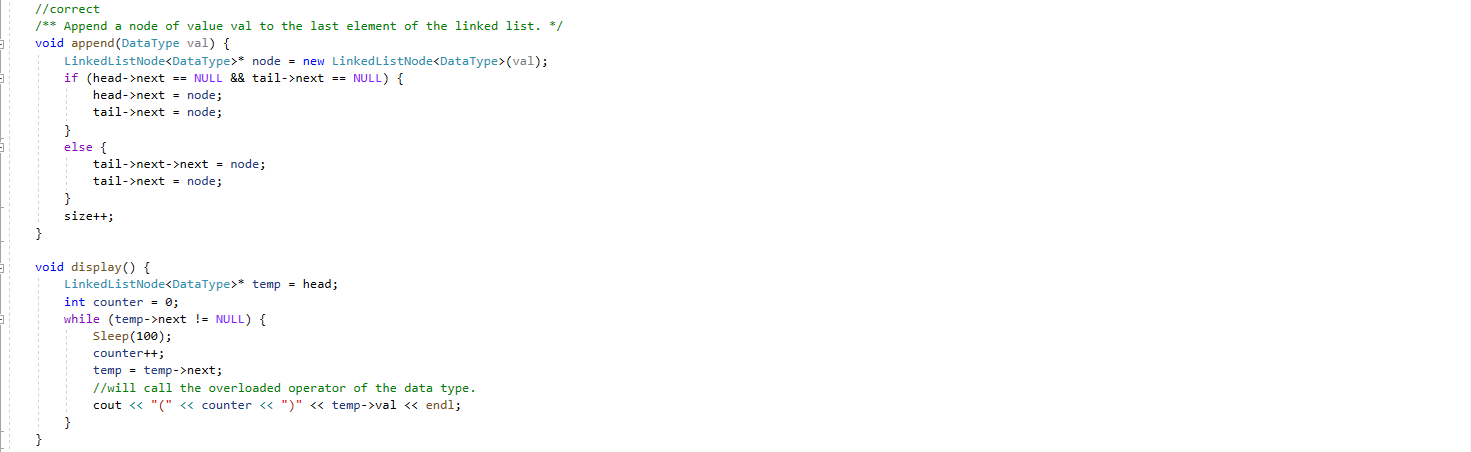
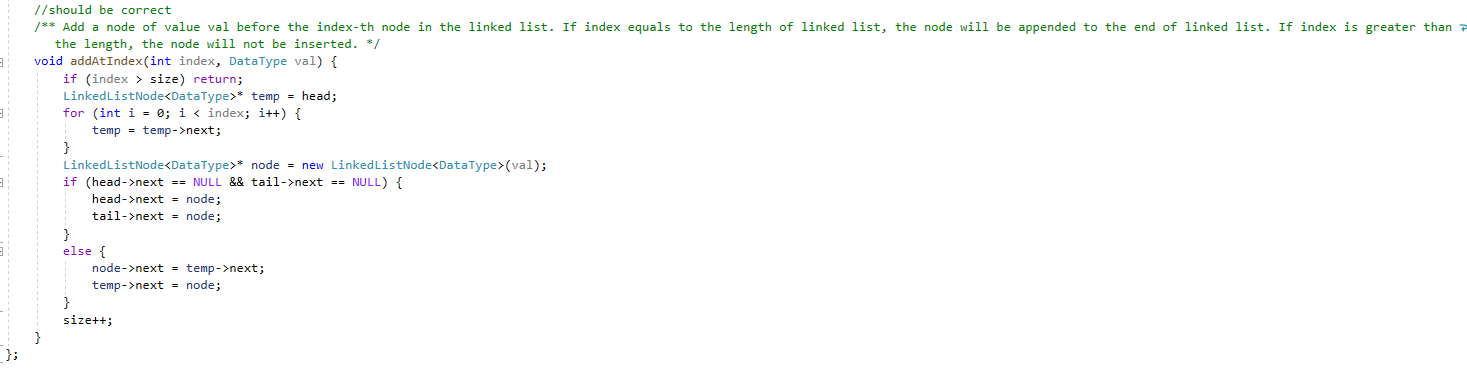
    

Figure: Linked list (LinkedList.h)

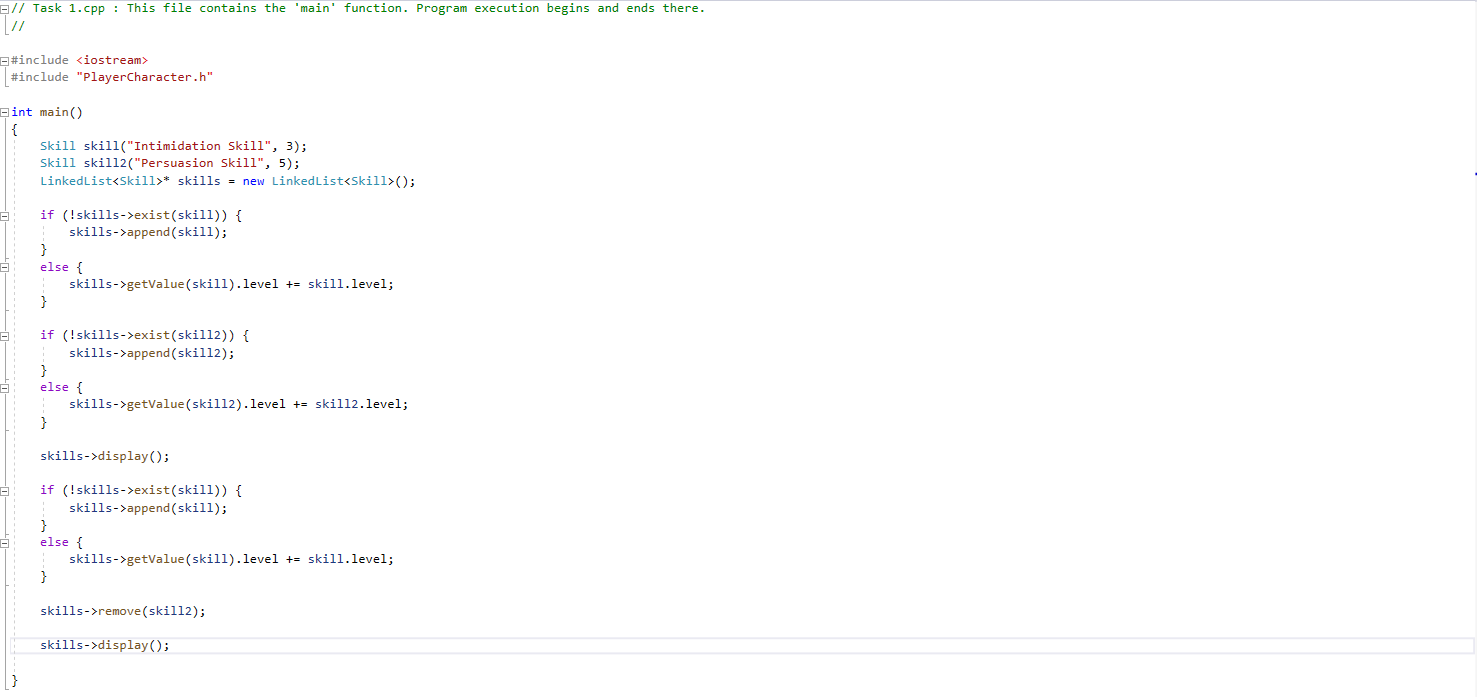


Figure: main (main.cpp)

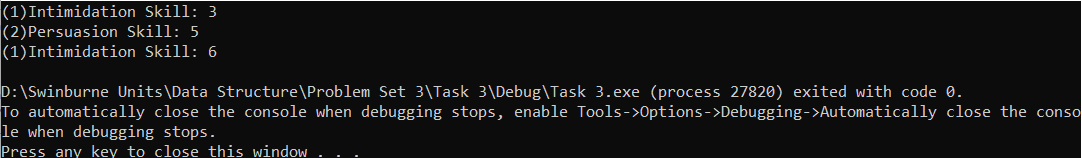


Figure: console output

**Troubleshooting**

There are certain functions of the linked list to search for an item. This could be done easily is the item is primitive data, but not composite data. For composite data, we need comparison operator overloading. Luckily, the compiler reports the problem clearly and I managed to find this solution on StackOverFlow.

**Task 4**

**Description**

Add a set of skills into the player character object at the start of the game. Implement the game loop to mimic a conversation. The response options should be available and will perform the necessary skill check. Then, correct feedback needs to be done based on the skill check.

**Concept**

This is to test the correctness of the linked list implementation and the control statement in the game loop. A check on the programmer’s cautiousness. The understanding of linked list function and manipulation of it in the functions of the player character class.

**Code**

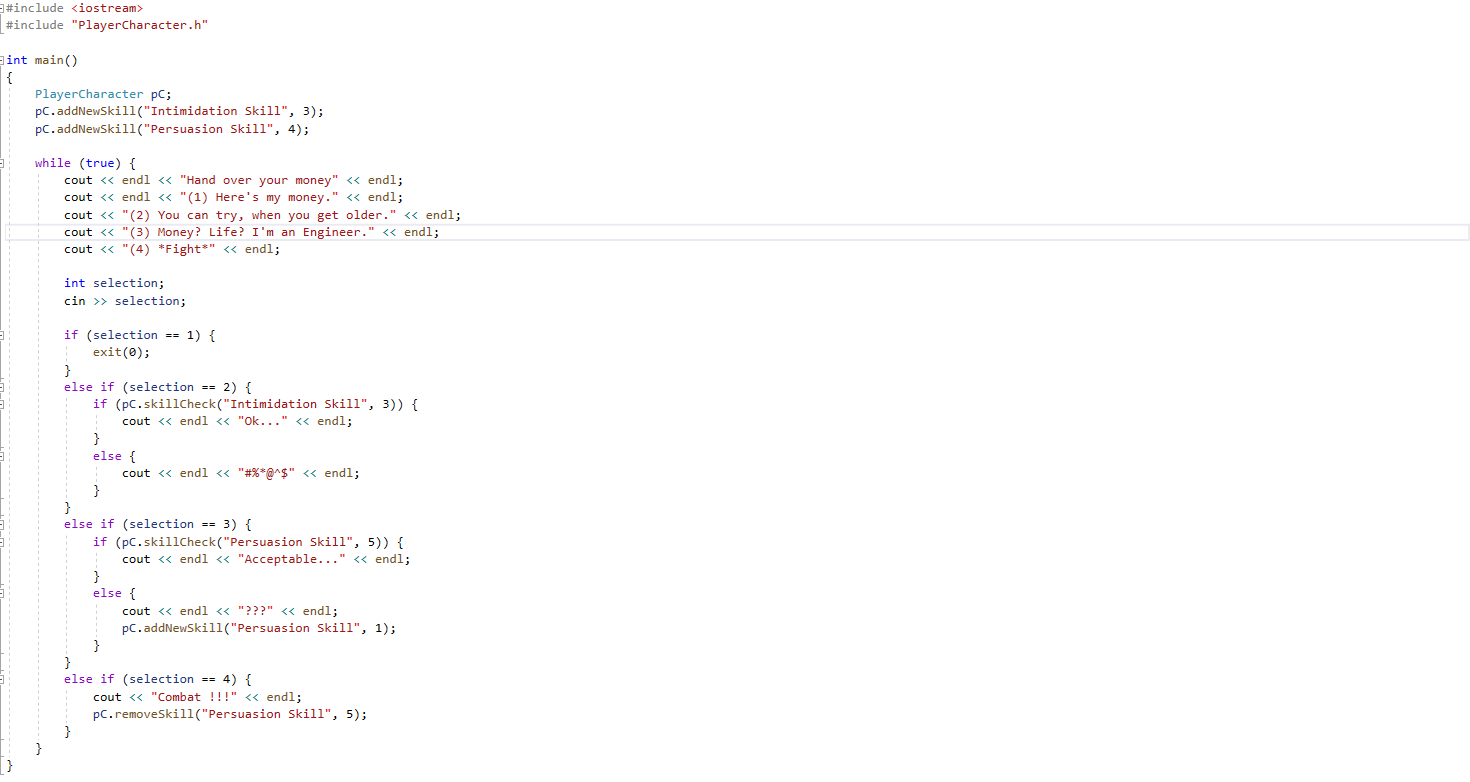
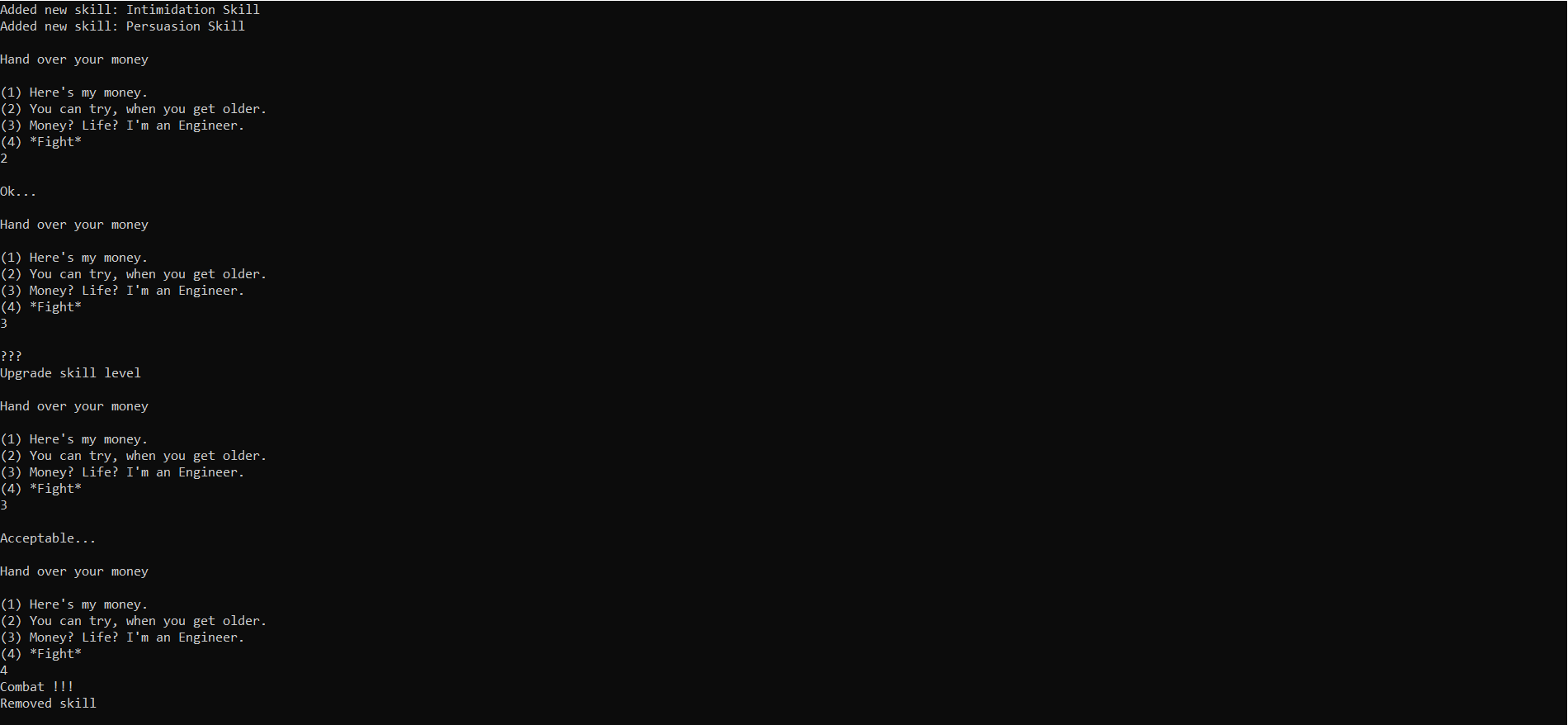
****

Figure: main function (main.cpp)



Figure: PlayerCharacter class (PlayerCharacter.h)

**Output**



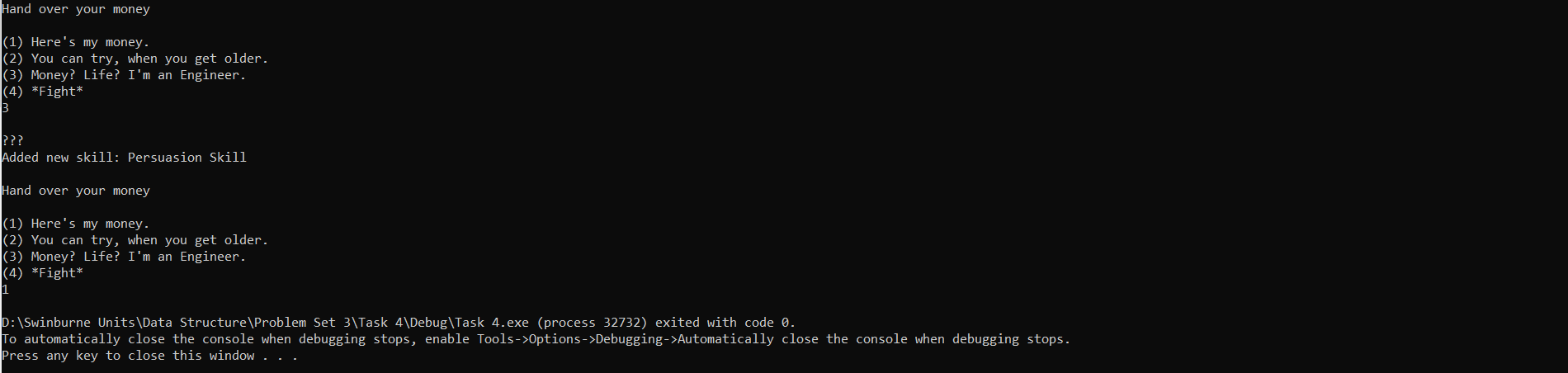


Figure: Console output

**Troubleshooting**

I must implement custom copy constructor to manage the dynamically allocated resource. Thanks to <https://stackoverflow.com/questions/3279543/what-is-the-copy-and-swap-idiom> , I managed to do it easily. The assignment operator could be implemented easily if the custom copy constructor had been implemented.

**Task 5**

**Description**

Review the ADT called “List” and its standard characteristics. Then, build the ADT and replace the current implementation in PlayerCharacter class with it.

**Concept**

The main motive is to accomplish the abstraction and encapsulation principles. Prior to the refactoring, the management code of linked list is hardcoded inside the character class. If we could move all these code into a separate class, then the code of character class will be cleaner and easier to modify. The character object only manipulates the linked list through the ADT functions.

**Implementation**

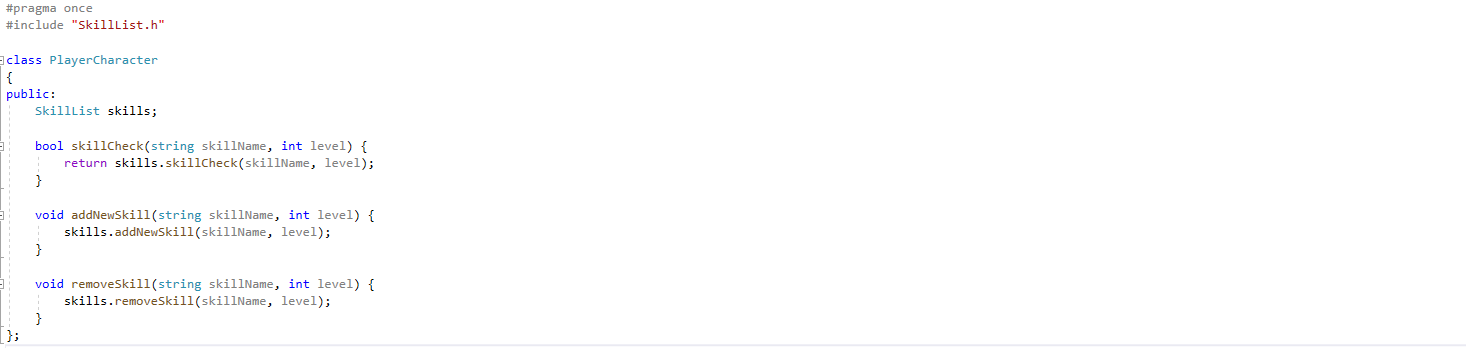
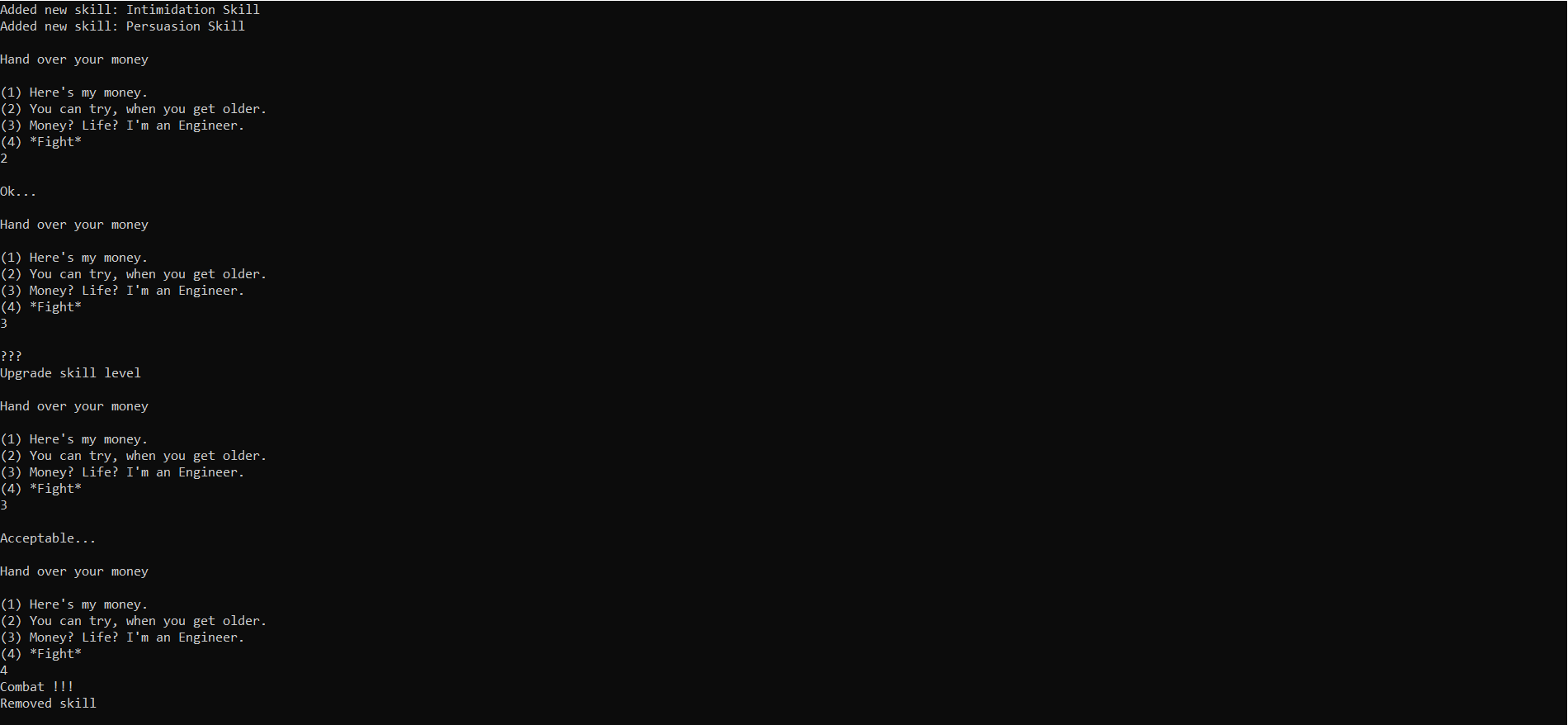


Figure: PlayerCharacter class (PlayerCharacter.h)



Figure: ADT (SkillList.h)

**Output**



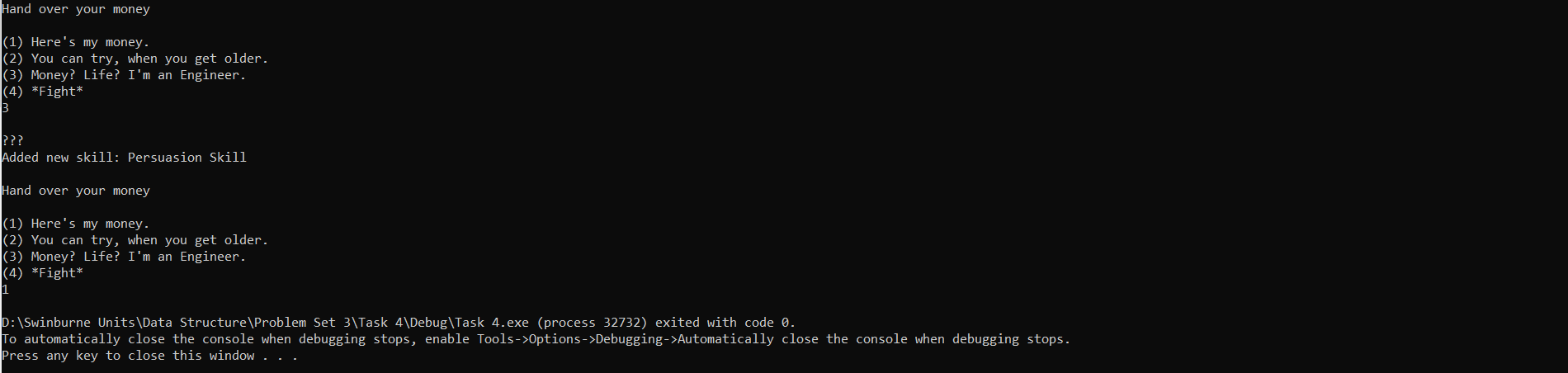


Figure: Console output

**Troubleshooting**

Fairly easy task. No problem.

**Appendix**



Figure: main.cpp

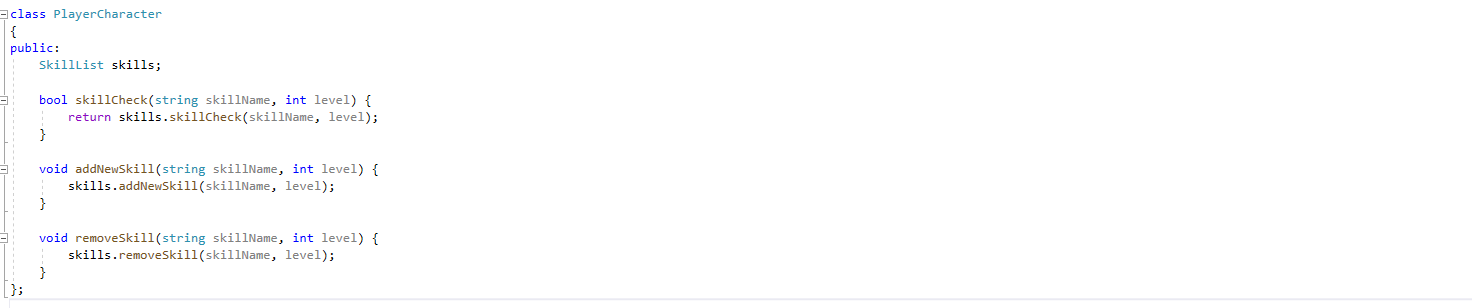
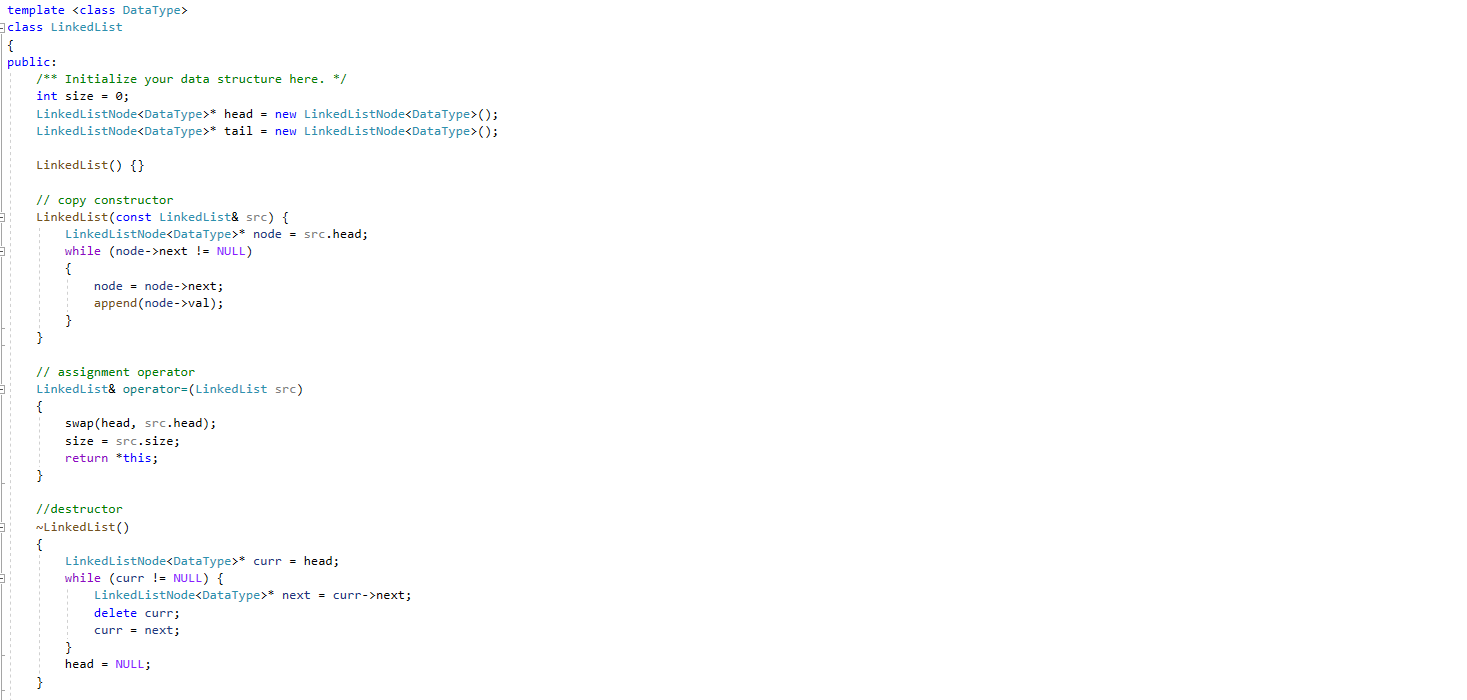
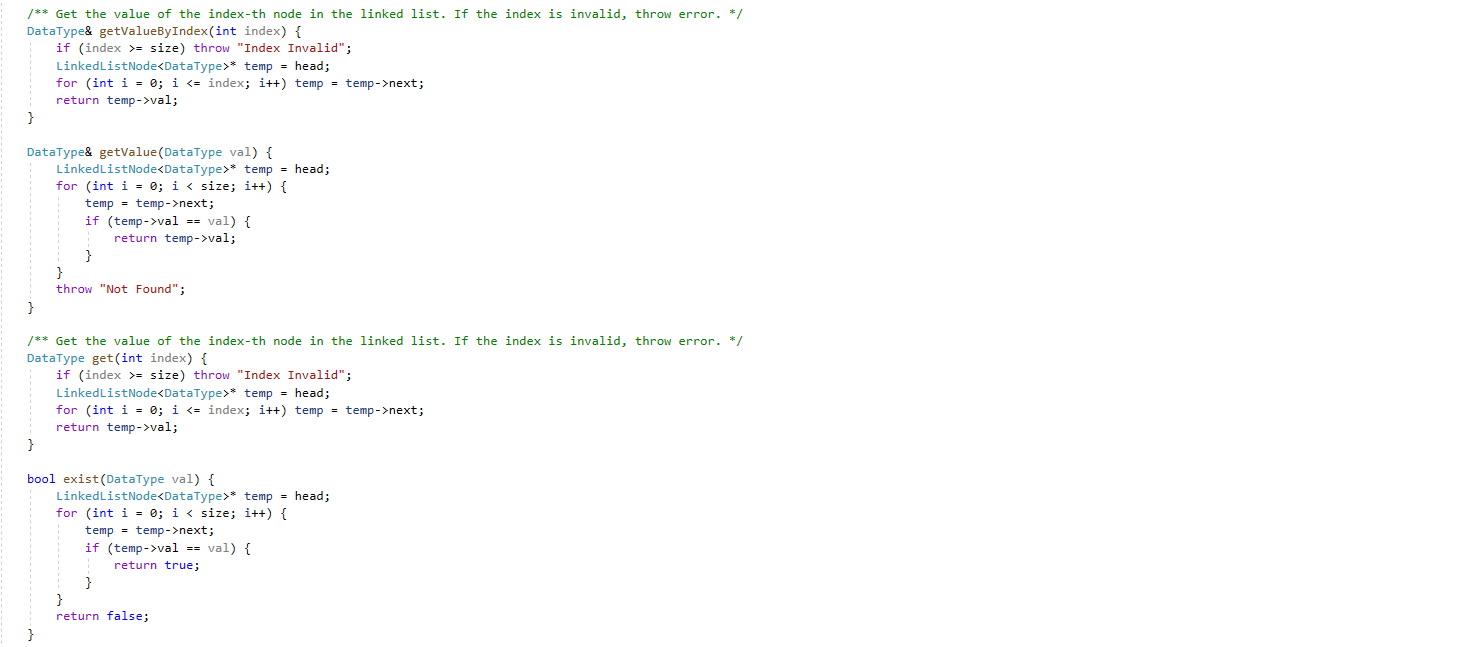


Figure: PlayerCharacter.h

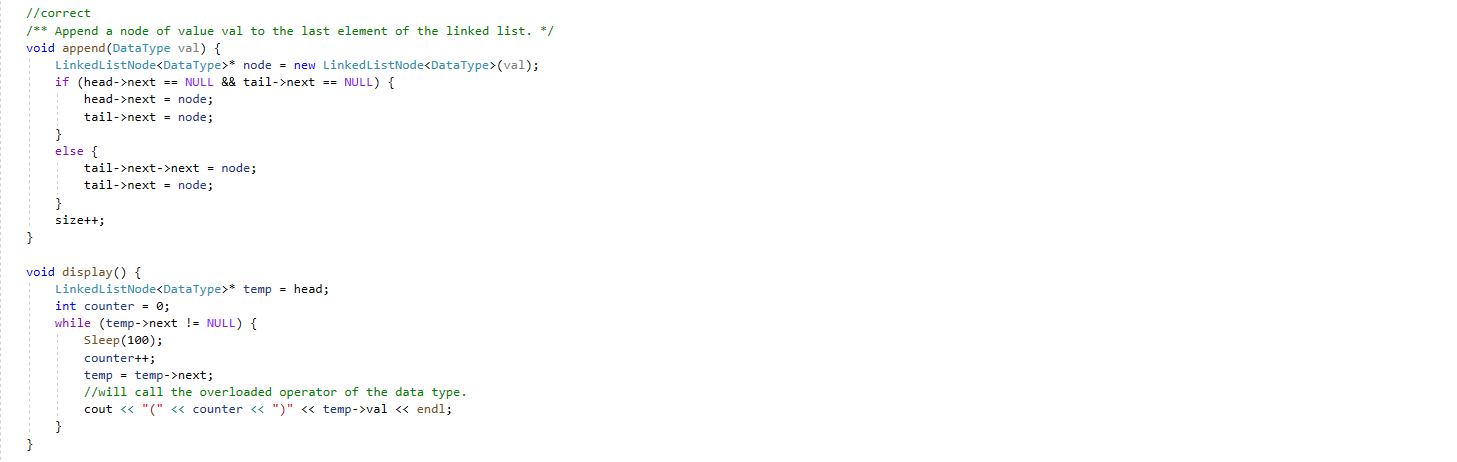


Figure: SkillList.h









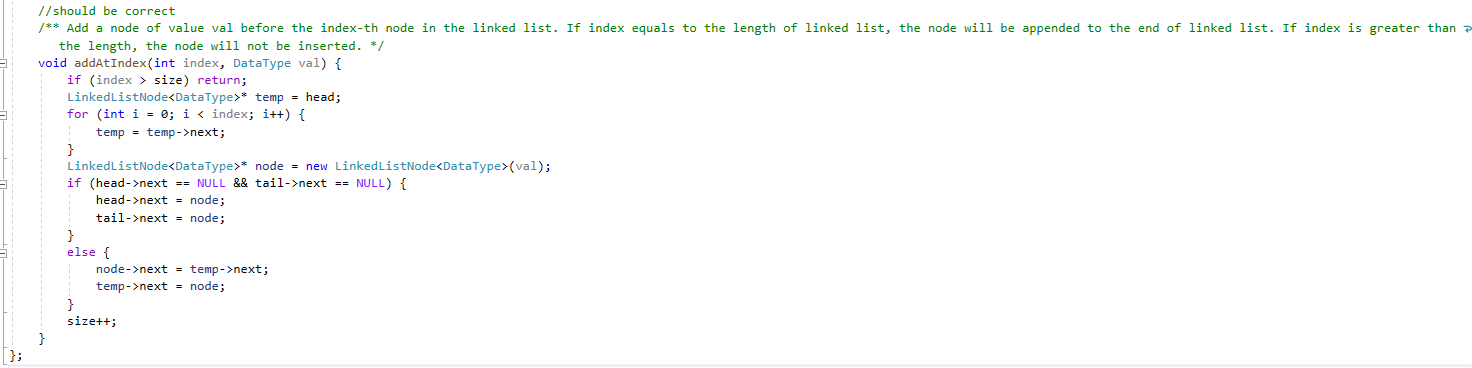


Figure: LinkedList.h



Figure: LinkedListNode.h