Assessment Cover Sheet

This Assessment Cover Sheet is only to be attached to hard copy submission of assessments.

ASSESSMENT DETAILS

Unit title	Data Structure and Pattern	Tutorial /Lab Group	1	Office use only
Unit code	COS30008	Due date	28 Oct 2022	
Name of lecturer/tutor	Dr. Mark Tee Kit Tsun			
Assignment title	Problem Set 3			Faculty or school date stamp

STUDENT(S) DETAILS

	Student Name(s)	Student ID Number(s)
(1)	Alex Ngie Guan Ming	102765770
(2)		
(3)		
(4)		
(5)		

DECLARATION AND STATEMENT OF AUTHORSHIP

1. I/we have not impersonated, or allowed myself/ourselves to be impersonated by any person for the purposes of this assessment.
2. This assessment is my/our original work and no part of it has been copied from any other source except where due acknowledgement is made.
3. No part of this assessment has been written for me/us by any other person except where such collaboration has been authorised by the lecturer/tutor concerned.
4. I/we have not previously submitted this work for this or any other course/unit.
5. I/we give permission for my/our assessment response to be reproduced, communicated, compared and archived for plagiarism detection, benchmarking or educational purposes.

I/we understand that:

6. Plagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a form of cheating and is a very serious academic offence that may lead to exclusion from the University. Plagiarised material can be drawn from, and presented in, written, graphic and visual form, including electronic data and oral presentations. Plagiarism occurs when the origin of the material used is not appropriately cited.

Student signature/s

I/we declare that I/we have read and understood the declaration and statement of authorship.

(1)	Alex	(4)	
(2)		(5)	
(3)		(6)	

Further information relating to the penalties for plagiarism, which range from a formal caution to expulsion from the University is contained on the Current Students website at **https://www.swinburne.edu.my/current-students/manage-course/exams-results-assessment**

Copies of this form can be downloaded from the Student Forms web page at **https://www.swinburne.edu.my/current-students/manage-course/exams-results-assessment/how-to-submit-work.php**

Contents

Task 1

Implementation

Task_1 — Character

```h
#pragma once
#include <iostream>
#include <string>

using namespace std;

class  Character {
private:
    string Name; // store the name of player
    int MaxHp; // store the max hp of player , current hp cnt more than max hp
    int CurrentHp; // store current hp of player
public:
    Character(string t_name,int t_hp); // constructor of character class
    int getMaxHp(); // return max hp of player
    int getCurrentHp(); // return current hp of player
    string getName(); // return name of the player
    void setCurrentHp(int dmg); // calculate current hp of player by passing income dmg
};
```

Task_1 — Character

```cpp
#include "Character.h"

Character::Character(string t_name,int t_hp) {
    Name = t_name;
    MaxHp = t_hp;
    CurrentHp = t_hp;
}

int Character::getMaxHp() {
    return MaxHp;
}

int Character::getCurrentHp() {
    return CurrentHp;
}

string Character::getName() {
    return Name;
}

void Character::setCurrentHp(int dmg) {
    CurrentHp -= dmg;
}
```

Task 2

Implementation

skillNode.h

```h
#pragma once
#include <iostream>
#include <string>

using namespace std;

class SkillNode {
private:
    string Name; // skill name
    int Level; // skill level
    SkillNode* next; // next node
    SkillNode* prev;// previous skill node
public:
    SkillNode(); //constructor of skill node
    SkillNode(string t_name,int t_level); // constructor of skill node with parameter
    int getLevel(); // return level
    string getName(); // return name
    SkillNode* getNextSkillNode(); // return next skill Node
    SkillNode* getPrevSkillNode(); // return prev skill Node
    void setNextSkillNode(SkillNode* temp); // set next skill node
    void setPrevSkillNode(SkillNode* temp); // set prev skill node
    void printDetail(); // print node detail
    void setLevel(int temp); // set node level
    void setName(string temp); // set node name
};
```

SkillNode.cpp

```cpp
Task_1 | SkillNode | getPrevSkillNode
#include "SkillNode.h"

SkillNode::SkillNode() {
    Name = "";
    Level = 0;
    next = NULL;
    prev = NULL;

}

SkillNode::SkillNode(string t_name,int t_level) {
    Name = t_name;
    Level = t_level;
    next = NULL;
    prev = NULL;
}

int SkillNode::getLevel() {
    return Level;
}

string SkillNode::getName() {
    return Name;
}

void SkillNode::setLevel(int temp) {
    Level = temp;
}

void SkillNode::setName(string temp) {
    Name = temp;
}
```

```
28  }
29
30  void SkillNode::setName(string temp) {
31      Name = temp;
32  }
33
34  SkillNode* SkillNode::getNextSkillNode() {
35      return next;
36  }
37
38  SkillNode* SkillNode::getPrevSkillNode() {
39      return prev;
40  }
41
42  void SkillNode::setNextSkillNode(SkillNode* temp) {
43      next = temp;
44  }
45
46  void SkillNode::setPrevSkillNode(SkillNode* temp) {
47      prev = temp;
48  }
49
50  void SkillNode::printDetail() {
51      cout << "Skill Detail" << endl;
52      cout << "Name : " << Name << endl;
53      cout << "Level : " << Level << endl << endl;
54  }
55
```

Task 3

Implementation

Character.h

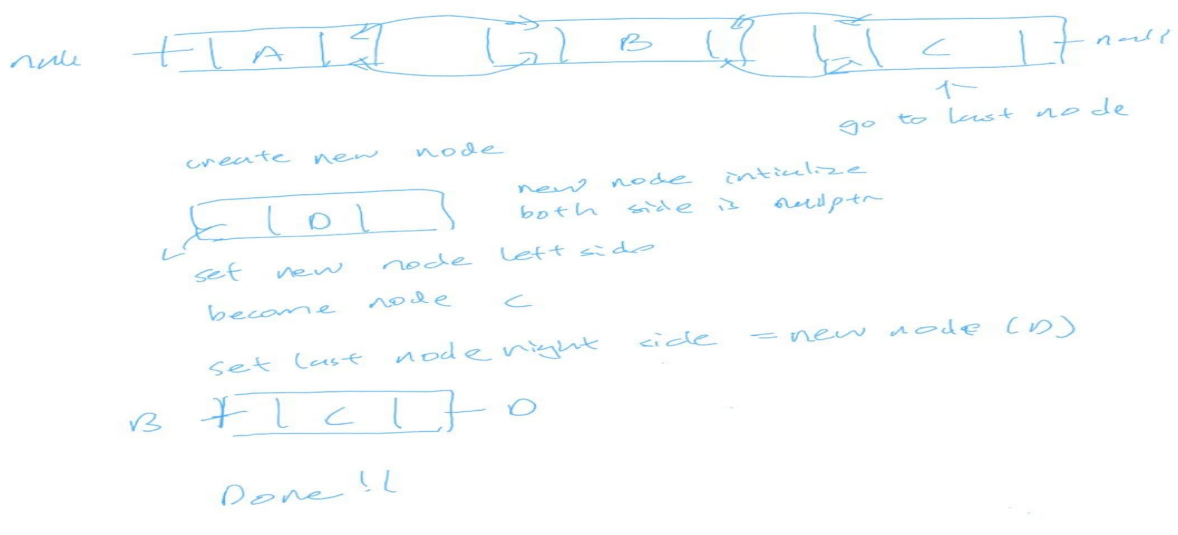
```
main.cpp | SkillNode.h | SkillNode.cpp | Character.h | Character.cpp
Task_1 | Character | PrintAllSkill()
```

```cpp
#pragma once
#include <iostream>
#include <string>
#include "SkillNode.h"


using namespace std;

class  Character {
private:
    string Name; // store the name of player
    int MaxHp; // store the max hp of player , current hp cnt more than max hp
    int CurrentHp; // store current hp of player
    //task 3
    SkillNode* skillNode; //pointerrrrrr of skill Node
public:
    Character(string t_name,int t_hp,SkillNode* s); // constructor of character class
    int getMaxHp(); // return max hp of player
    int getCurrentHp(); // return current hp of player
    string getName(); // return name of the player
    void setCurrentHp(int dmg); // calculate current hp of player by passing income dmg
    //task 3
    void find(string s_name); //task 3 - find skill exist or not
    void add(string s_temp, int s_level); // task 3 - add new skill node
    void remove(string s_name); // task 3 - remove skill node
    void read(); // task 3 - read the content of skill node
    void modify(string temp,string s_name, int s_level); // task 3 - modfify the content of skill node
    void PrintAllSkill(); //
};
```

Character.cpp

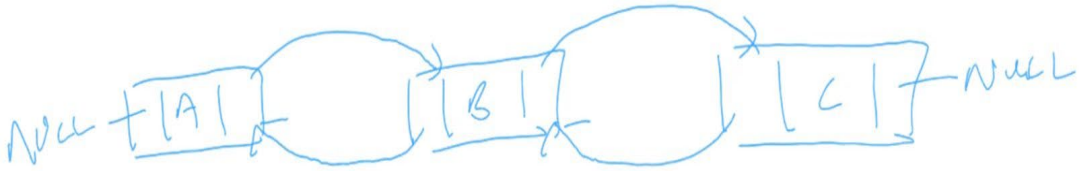
Add

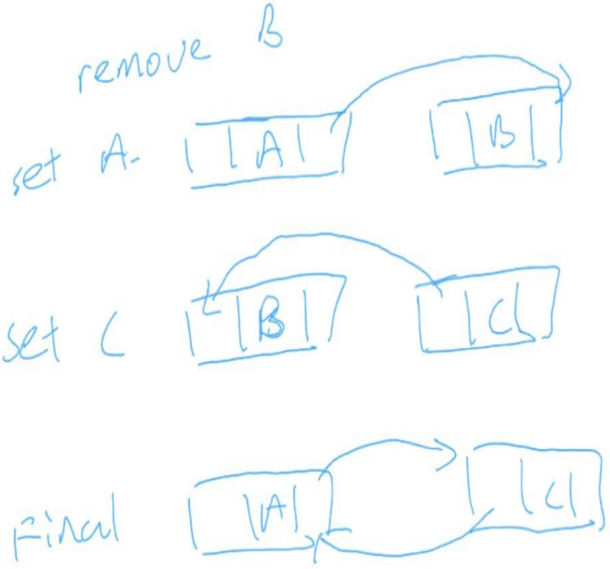
```cpp
void Character::add(string s_temp,int s_level) { // add new skill node
    SkillNode* t_skillNode = *&skillNode; //copy pointer reference
    if (t_skillNode == nullptr) { // if skill node is empty
        skillNode = new SkillNode(s_temp, s_level); // this skill node constructor already initialize prev and next node is null
    }
    else { // if skill node exist something
        while (t_skillNode->getNextSkillNode() != NULL) { // loop until the last skill node
            t_skillNode = t_skillNode->getNextSkillNode();
        }
        SkillNode* temp = new SkillNode(s_temp, s_level); // create new skill node want tp add
        temp->setPrevSkillNode(t_skillNode); // set new skill node prev skill node
        t_skillNode->setNextSkillNode(temp); // insert new skill node to the last skill node

    }
    cout << "\n Successfully create new skill \n" << endl;
}
```



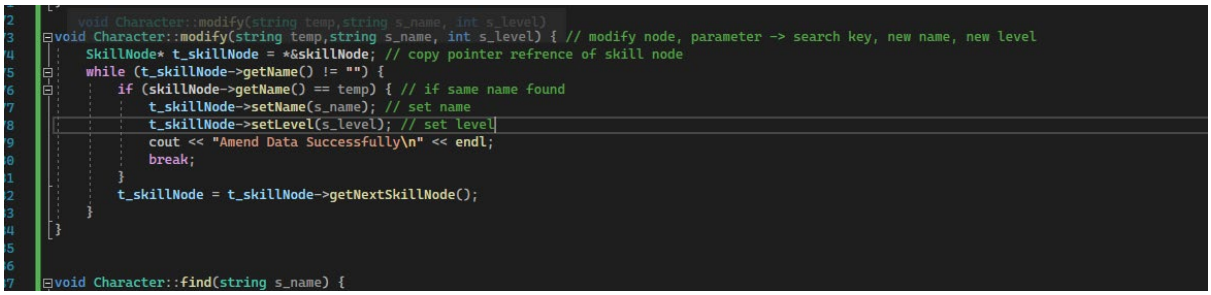
Remove

```
41
42  void Character::remove(string s_name) {
43      SkillNode* t_skillNode = *&skillNode; // create pointer refrence of skill node
44      while (t_skillNode->getName() != "") { // check if not empty
45          if (t_skillNode->getName() == s_name) { // check if name same with selected name
46              if (t_skillNode->getNextSkillNode() == nullptr && t_skillNode->getPrevSkillNode() == nullptr) { //if current node both side (prev,next) is null
47                  skillNode = new SkillNode(); // skill node init to empty
48              }
49              else if (t_skillNode->getPrevSkillNode() == nullptr && t_skillNode->getNextSkillNode() != nullptr) { // if prev is null and next exist, first node
50                  t_skillNode = t_skillNode->getNextSkillNode(); // get next skill node
51                  t_skillNode->setPrevSkillNode(nullptr); //set this node prev is null
52                  skillNode = t_skillNode; // reassign to skill node, now the node become header
53              }
54              else { // below means selected node have previos node
55                  if (t_skillNode->getNextSkillNode() != nullptr) { // if next node exist
56                      t_skillNode->getNextSkillNode()->setPrevSkillNode(t_skillNode->getNextSkillNode()); // set current node's prev node is current node's next node
57                      t_skillNode->getPrevSkillNode()->setNextSkillNode(t_skillNode->getPrevSkillNode()); // set current node's next node is current node's prev node
58                  }
59                  else { // if next node no exist , (in last node)
60                      t_skillNode->getPrevSkillNode()->setNextSkillNode(nullptr);
61                      // set next node to nullptr , this mean current node is last node, so no need to change prev node
62                  }
63              }
64              cout << t_skillNode->getName() << " remove Successfully\n" << endl;
65              read(); // print current double link-list
66              break; // stop while loop and quit
67
68          }
69          t_skillNode = t_skillNode->getNextSkillNode(); // go to next node, if not found same skill name
70      }
71  }
72
```



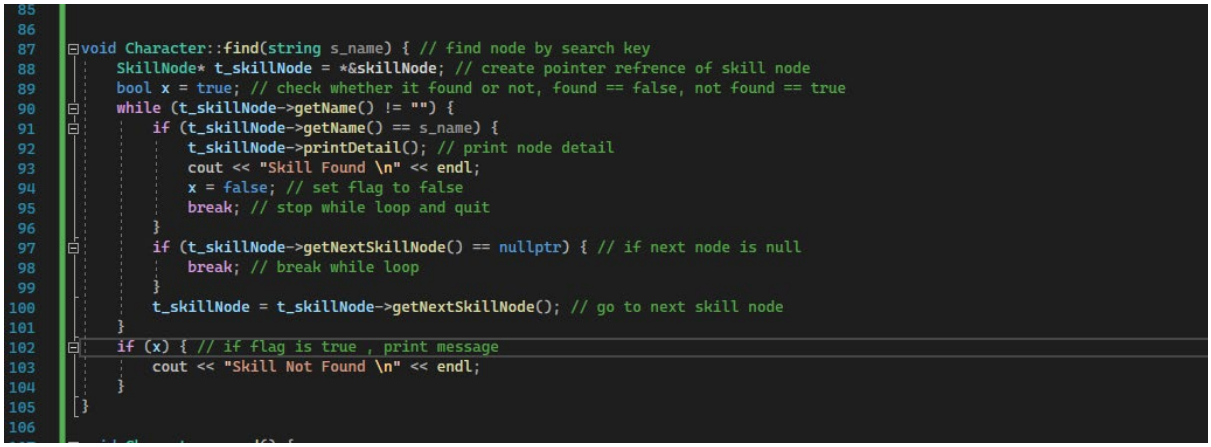


Modify



```
void Character::modify(string temp,string s_name, int s_level) { // modify node, parameter -> search key, new name, new level
    SkillNode* t_skillNode = *&skillNode; // copy pointer refrence of skill node
    while (t_skillNode->getName() != "") {
        if (skillNode->getName() == temp) { // if same name found
            t_skillNode->setName(s_name); // set name
            t_skillNode->setLevel(s_level); // set level
            cout << "Amend Data Successfully\n" << endl;
            break;
        }
        t_skillNode = t_skillNode->getNextSkillNode();
    }
}


void Character::find(string s_name) {
```

Find

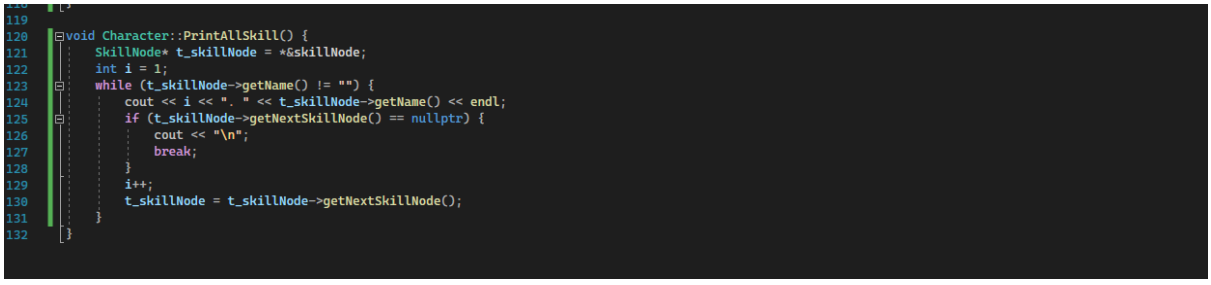


```
void Character::find(string s_name) { // find node by search key
    SkillNode* t_skillNode = *&skillNode; // create pointer refrence of skill node
    bool x = true; // check whether it found or not, found == false, not found == true
    while (t_skillNode->getName() != "") {
        if (t_skillNode->getName() == s_name) {
            t_skillNode->printDetail(); // print node detail
            cout << "Skill Found \n" << endl;
            x = false; // set flag to false
            break; // stop while loop and quit
        }
        if (t_skillNode->getNextSkillNode() == nullptr) { // if next node is null
            break; // break while loop
        }
        t_skillNode = t_skillNode->getNextSkillNode(); // go to next skill node
    }
    if (x) { // if flag is true , print message
        cout << "Skill Not Found \n" << endl;
    }
}
```

Read

```
void Character::read() {
    SkillNode* t_skillNode = *&skillNode;
    while (t_skillNode->getName() != "") {
        cout << t_skillNode->getName();
        if (t_skillNode->getNextSkillNode() == nullptr) {
            cout << "\n";
            break;
        }
        cout << " -> ";
        t_skillNode = t_skillNode->getNextSkillNode();
    }
}
```

Other function created

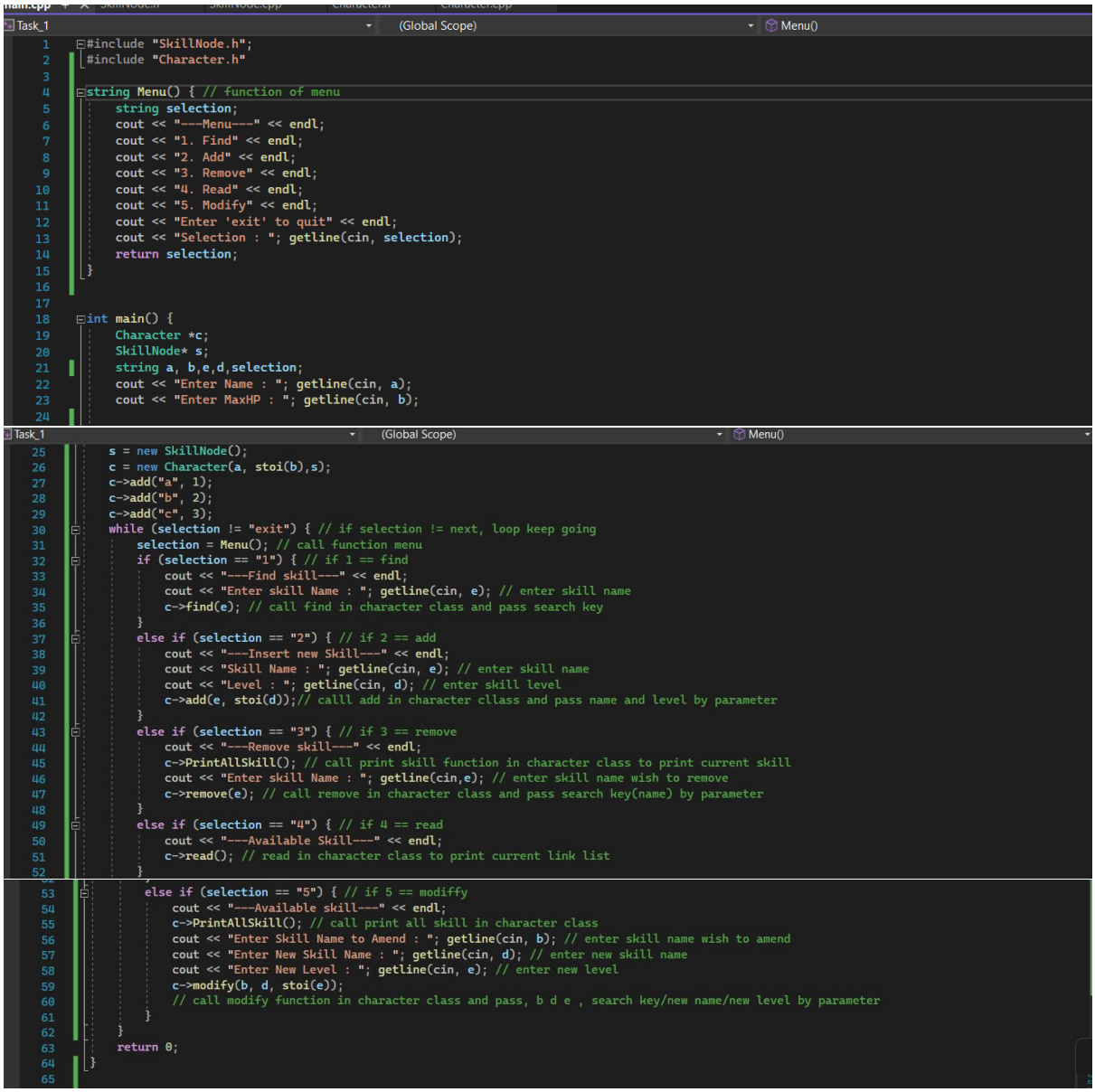


```
void Character::PrintAllSkill() {
    SkillNode* t_skillNode = *&skillNode;
    int i = 1;
    while (t_skillNode->getName() != "") {
        cout << i << ". " << t_skillNode->getName() << endl;
        if (t_skillNode->getNextSkillNode() == nullptr) {
            cout << "\n";
            break;
        }
        i++;
        t_skillNode = t_skillNode->getNextSkillNode();
    }
}
```

Task 4

Implementation

Main.cpp

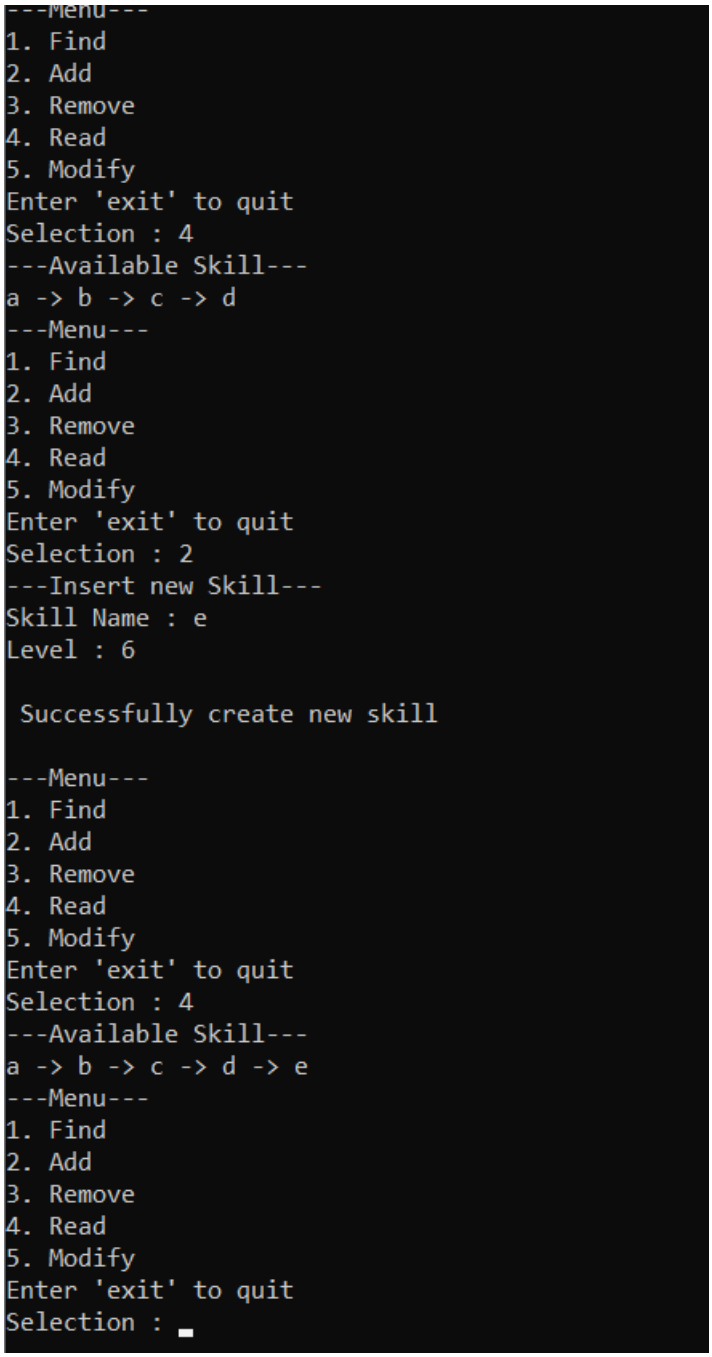


```cpp
#include "SkillNode.h";
#include "Character.h"

string Menu() { // function of menu
    string selection;
    cout << "---Menu---" << endl;
    cout << "1. Find" << endl;
    cout << "2. Add" << endl;
    cout << "3. Remove" << endl;
    cout << "4. Read" << endl;
    cout << "5. Modify" << endl;
    cout << "Enter 'exit' to quit" << endl;
    cout << "Selection : "; getline(cin, selection);
    return selection;
}


int main() {
    Character *c;
    SkillNode* s;
    string a, b,e,d,selection;
    cout << "Enter Name : "; getline(cin, a);
    cout << "Enter MaxHP : "; getline(cin, b);

    s = new SkillNode();
    c = new Character(a, stoi(b),s);
    c->add("a", 1);
    c->add("b", 2);
    c->add("c", 3);
    while (selection != "exit") { // if selection != next, loop keep going
        selection = Menu(); // call function menu
        if (selection == "1") { // if 1 == find
            cout << "---Find skill---" << endl;
            cout << "Enter skill Name : "; getline(cin, e); // enter skill name
            c->find(e); // call find in character class and pass search key
        }
        else if (selection == "2") { // if 2 == add
            cout << "---Insert new Skill---" << endl;
            cout << "Skill Name : "; getline(cin, e); // enter skill name
            cout << "Level : "; getline(cin, d); // enter skill level
            c->add(e, stoi(d));// calll add in character cllass and pass name and level by parameter
        }
        else if (selection == "3") { // if 3 == remove
            cout << "---Remove skill---" << endl;
            c->PrintAllSkill(); // call print skill function in character class to print current skill
            cout << "Enter skill Name : "; getline(cin,e); // enter skill name wish to remove
            c->remove(e); // call remove in character class and pass search key(name) by parameter
        }
        else if (selection == "4") { // if 4 == read
            cout << "---Available Skill---" << endl;
            c->read(); // read in character class to print current link list
        }
        else if (selection == "5") { // if 5 == modiffy
            cout << "---Available skill---" << endl;
            c->PrintAllSkill(); // call print all skill in character class
            cout << "Enter Skill Name to Amend : "; getline(cin, b); // enter skill name wish to amend
            cout << "Enter New Skill Name : "; getline(cin, d); // enter new skill name
            cout << "Enter New Level : "; getline(cin, e); // enter new level
            c->modify(b, d, stoi(e));
            // call modify function in character class and pass, b d e , search key/new name/new level by parameter
        }
    }
    return 0;
}
```

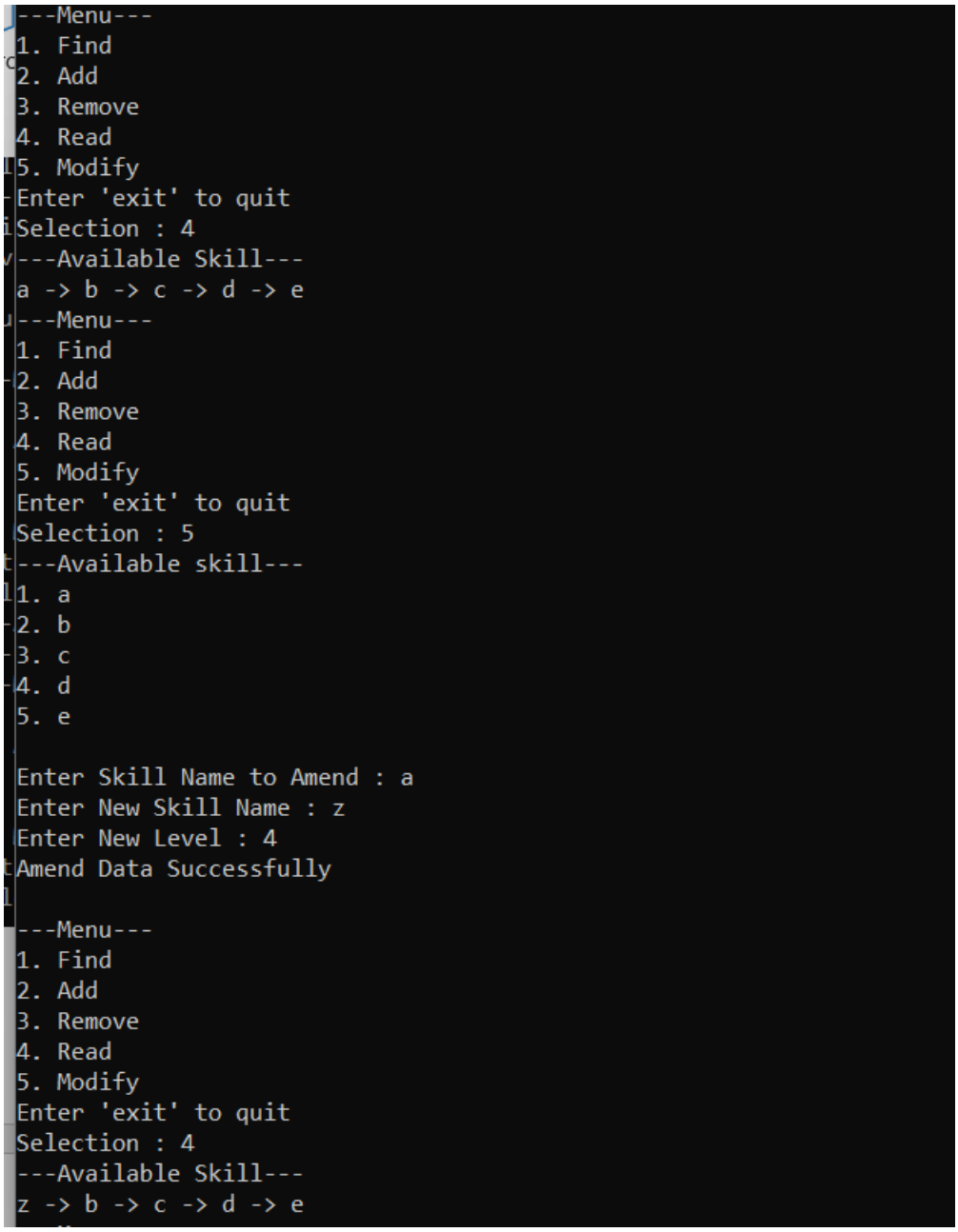
Output

Find

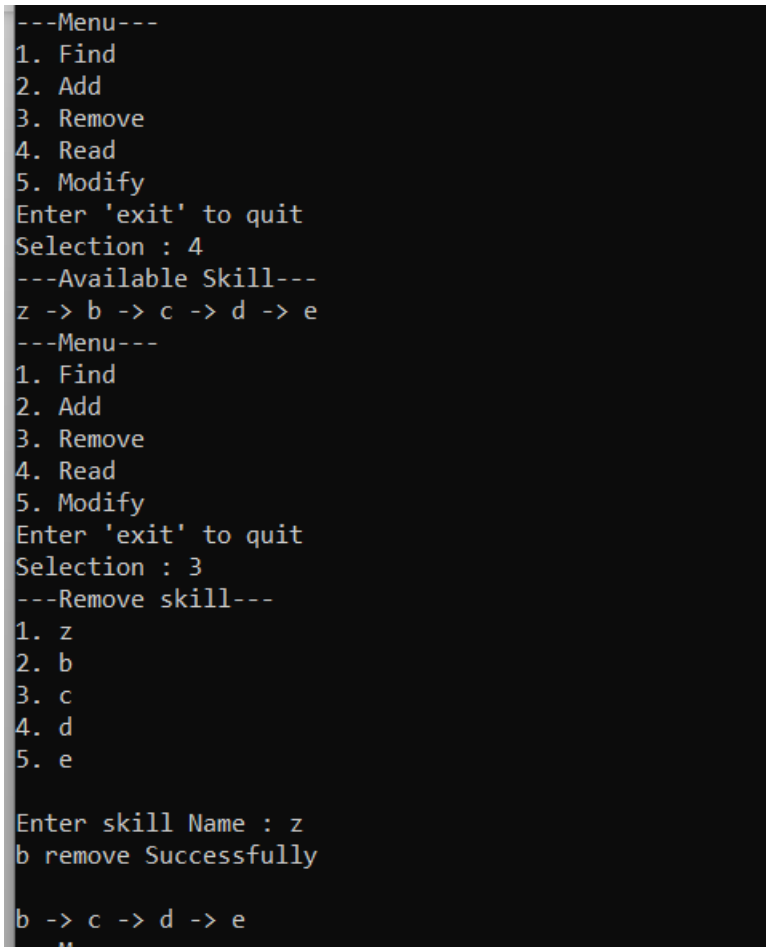
Read and Add



Modify


---Menu---
1. Find
2. Add
3. Remove
4. Read
5. Modify
Enter 'exit' to quit
Selection : 4
---Available Skill---
a -> b -> c -> d -> e
---Menu---
1. Find
2. Add
3. Remove
4. Read
5. Modify
Enter 'exit' to quit
Selection : 5
---Available skill---
1. a
2. b
3. c
4. d
5. e
Enter Skill Name to Amend : a
Enter New Skill Name : z
Enter New Level : 4
Amend Data Successfully
---Menu---
1. Find
2. Add
3. Remove
4. Read
5. Modify
Enter 'exit' to quit
Selection : 4
---Available Skill---
z -> b -> c -> d -> e

Remove

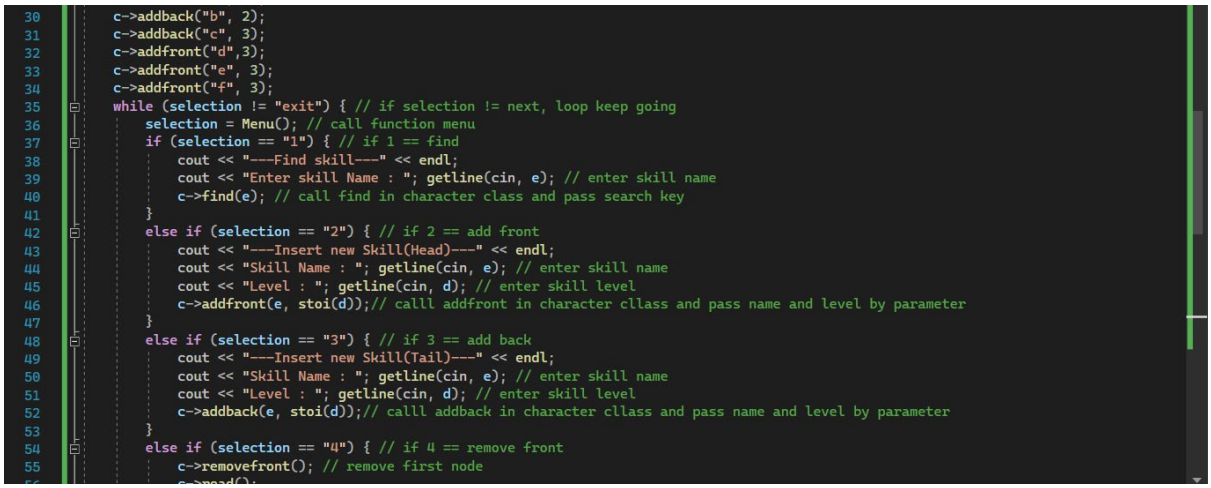

---Menu---
1. Find
2. Add
3. Remove
4. Read
5. Modify
Enter 'exit' to quit
Selection : 4
---Available Skill---
z -> b -> c -> d -> e
---Menu---
1. Find
2. Add
3. Remove
4. Read
5. Modify
Enter 'exit' to quit
Selection : 3
---Remove skill---
1. z
2. b
3. c
4. d
5. e
Enter skill Name : z
b remove Successfully
b -> c -> d -> e

Task 5

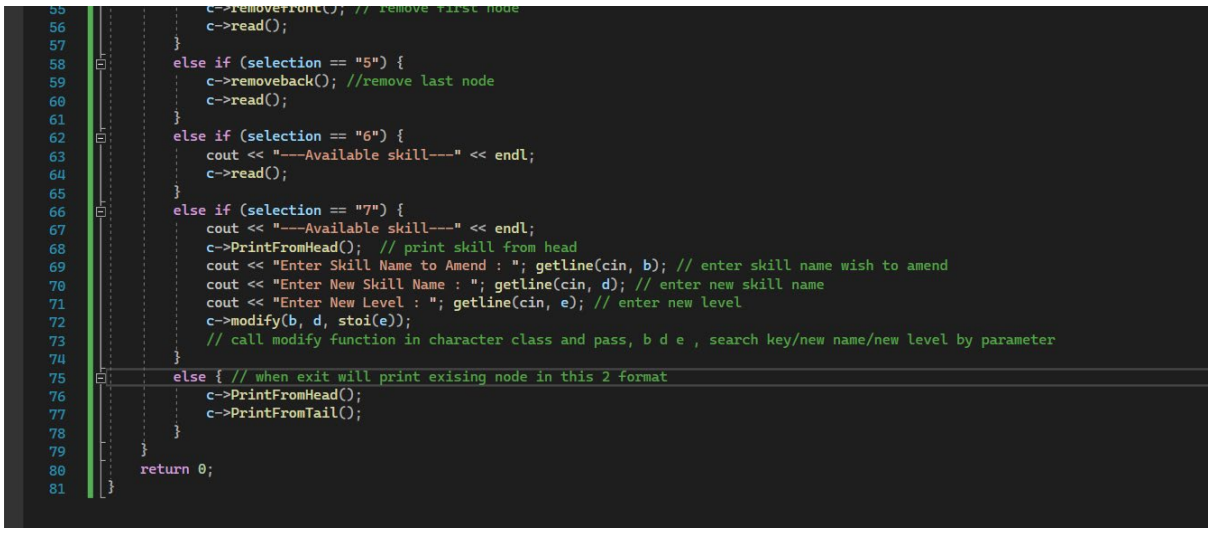
Implementation

Main.cpp

```cpp
#include "Character.h"
#include "SkillList.h"

string Menu() { // function of menu
    string selection;
    cout << "---Menu---" << endl;
    cout << "1. Find" << endl;
    cout << "2. Add head" << endl;
    cout << "3. Add tail" << endl;
    cout << "4. Remove head" << endl;
    cout << "5. Remove Tail" << endl;
    cout << "6. Read" << endl;
    cout << "7. Modify" << endl;
    cout << "Enter 'exit' to quit" << endl;
    cout << "Selection : "; getline(cin, selection);
    return selection;
}


int main() {
    Character* c;
    SkillNode* s;
    string a, b, e, d, selection;
    cout << "Enter Name : "; getline(cin, a);
    cout << "Enter MaxHP : "; getline(cin, b);

    s = new SkillNode();
    c = new Character(a, stoi(b));
    c->addback("a", 1);
```



```cpp
    c->addback("b", 2);
    c->addback("c", 3);
    c->addfront("d",3);
    c->addfront("e", 3);
    c->addfront("f", 3);
    while (selection != "exit") { // if selection != next, loop keep going
        selection = Menu(); // call function menu
        if (selection == "1") { // if 1 == find
            cout << "---Find skill---" << endl;
            cout << "Enter skill Name : "; getline(cin, e); // enter skill name
            c->find(e); // call find in character class and pass search key
        }
        else if (selection == "2") { // if 2 == add front
            cout << "---Insert new Skill(Head)---" << endl;
            cout << "Skill Name : "; getline(cin, e); // enter skill name
            cout << "Level : "; getline(cin, d); // enter skill level
            c->addfront(e, stoi(d));// calll addfront in character cllass and pass name and level by parameter
        }
        else if (selection == "3") { // if 3 == add back
            cout << "---Insert new Skill(Tail)---" << endl;
            cout << "Skill Name : "; getline(cin, e); // enter skill name
            cout << "Level : "; getline(cin, d); // enter skill level
            c->addback(e, stoi(d));// calll addback in character cllass and pass name and level by parameter
        }
        else if (selection == "4") { // if 4 == remove front
            c->removefront(); // remove first node
```



```cpp
            c->read();
        }
        else if (selection == "5") {
            c->removeback(); //remove last node
            c->read();
        }
        else if (selection == "6") {
            cout << "---Available skill---" << endl;
            c->read();
        }
        else if (selection == "7") {
            cout << "---Available skill---" << endl;
            c->PrintFromHead();  // print skill from head
            cout << "Enter Skill Name to Amend : "; getline(cin, b); // enter skill name wish to amend
            cout << "Enter New Skill Name : "; getline(cin, d); // enter new skill name
            cout << "Enter New Level : "; getline(cin, e); // enter new level
            c->modify(b, d, stoi(e));
            // call modify function in character class and pass, b d e , search key/new name/new level by parameter
        }
        else { // when exit will print exising node in this 2 format
            c->PrintFromHead();
            c->PrintFromTail();
        }
    }
    return 0;
}
```

Character.h

```h
#pragma once
#include <iostream>
#include <string>
#include "SkillList.h"

using namespace std;

class  Character {
private:
    string Name; // store the name of player
    int MaxHp; // store the max hp of player , current hp cnt more than max hp
    int CurrentHp; // store current hp of player
    SkillList SL;
public:
    Character(string t_name, int t_hp); // constructor of character class
    int getMaxHp(); // return max hp of player
    int getCurrentHp(); // return current hp of player
    string getName(); // return name of the player
    void setCurrentHp(int dmg); // calculate current hp of player by passing income dmg
    //task 3
    void find(string s_name); //task 3 - find skill exist or not
    void addfront(string s_temp, int s_level); // task 3 - insert node at front
    void addback(string s_temp, int s_level); // task 3 - insert node at back
    void removefront(); // task 3 - remove skill node
    void removeback(); // task 3 - remove skill node
    void read(); // task 3 - read the content of skill node
    void modify(string temp, string s_name, int s_level); // task 3 - modfify the content of skill node
    void PrintFromTail();
    void PrintFromHead();
};

```

Character.cpp

```cpp
void Character::addfront(string s_temp, int s_level) { // call skilllist add front
    SL.addfront(s_temp,s_level);
}

void Character::addback(string s_temp, int s_level) {// call skilllist add back function
    SL.addback(s_temp, s_level);
}

void Character::removefront() { // call skill list remove front function
    SL.removefront();
}

void Character::removeback() { //call skill list remove back function
    SL.removeback();
}

void Character::modify(string temp, string s_name, int s_level) { // call skill list modify function
    SL.modify(temp, s_name, s_level);
}


void Character::find(string s_name) { // call skill list find function
    SL.find(s_name);
}
```

```cpp
void Character::read() { // call skill list read function
    SL.read();
}

void Character::PrintFromTail() { // call skill list read from tail function
    SL.PrintFromTail();
}

void Character::PrintFromHead() {// call skill list read from head function
    SL.PrintFromHead();
}
```

SkillList.h

```h
#pragma once
#include <iostream>
#include <string>
#include "SkillNode.h"

using namespace std;

class SkillList {
private:
    SkillNode* skillNode;
    SkillNode* head;
    SkillNode* tail;
public:
    SkillList();
    void find(string s_name); //task 5 - find skill exist or not
    void addfront(string s_temp, int s_level); // task 5 - add new skill node at first
    void addback(string s_temp, int s_level); // task 5 - add new skill node at last
    void removefront(); // task 5 - remove frsit skill node
    void removeback(); // task 5 - remove last skill node
    void read(); // task 5 - read the content of skill node
    void modify(string temp, string s_name, int s_level); // task 3 - modfify the content of skill node
    void PrintFromTail(); //
    void PrintFromHead();
};
```

SkillList.cpp

Add back

```cpp
#include "SkillList.h"

SkillList::SkillList() {
    head = NULL;
    tail = NULL;
    skillNode = NULL;
}

void SkillList::addback(string s_temp, int s_level) { // add new skill node
    SkillNode* current_tail = *&tail; //create pointer reference of skillnode
    if (current_tail == nullptr) { // if skill node is empty
        skillNode = new SkillNode(s_temp, s_level); // this skill node constructor already initialize prev and next node is null
        head = *&skillNode;// become pointer refrence of skill node
        tail = *&skillNode;// become pointer refrence of skill node
    }
    else { // if skill node exist something
        SkillNode* temp = new SkillNode(s_temp, s_level); // create new skill node want tp add
        temp->setPrevSkillNode(current_tail); // set previous node(new node) is current tail
        current_tail->setNextSkillNode(temp); // set current tail next node is new node
        tail = temp; // new node become tail
    }
    cout << "\n Successfully add new skill at back\n" << endl;
}
```

Add front / remove front

```cpp
void SkillList::addfront(string s_temp, int s_level) {
    SkillNode* current_head = *&head; //create pointer reference of skillnode
    if (current_head == nullptr) { // if skill node is empty
        skillNode = new SkillNode(s_temp, s_level); // this skill node constructor already initialize prev and next node is null
        head = *&skillNode; // become pointer refrence of skill node
        tail = *&skillNode; // become pointer refrence of skill node
    }
    else { // if skill node exist something
        SkillNode* temp = new SkillNode(s_temp, s_level); // create new skill node want to add
        temp->setNextSkillNode(current_head); // set next skill node(new node) is current head
        current_head->setPrevSkillNode(temp); // set current head prev skill node is new skill node
        head = temp; // new skill node become head
    }
    cout << "\n Successfully add new skill at front\n" << endl;
}

void SkillList::removefront() {
    SkillNode* current_head = *&head; // get current head
    if (current_head->getNextSkillNode() != nullptr) { // when more that 2 node
        SkillNode* temp = current_head->getNextSkillNode(); //save current head next node to temp
        temp->setPrevSkillNode(nullptr); // set temp previous node to null
        head = temp; // temp become head
    }
    else { // only 1 node
        skillNode = nullptr;
        head = nullptr;
        tail = nullptr;
    }
}
```

Remove back/ modify

```cpp
void SkillList::removeback() {// when more that 2 node
    SkillNode* current_tail = *&tail; // get current tail
    if (current_tail->getPrevSkillNode() != nullptr) {// when more that 2 node
        SkillNode* temp = current_tail->getPrevSkillNode(); // save current tail prev node to temp
        temp->setNextSkillNode(nullptr); // set temp next node to null
        tail = temp; // temp become tail
    }
    else {// only 1 node
        skillNode = nullptr;
        head = nullptr;
        tail = nullptr;
    }
}

void SkillList::modify(string temp, string s_name, int s_level) { // modify node, parameter -> search key, new name, new level
    SkillNode* t_skillNode = *&skillNode; // copy pointer refrence
    while (t_skillNode->getName() != "") {
        if (skillNode->getName() == temp) { // if same name found
            t_skillNode->setName(s_name); // set name
            t_skillNode->setLevel(s_level); // set level
            cout << "Amend Data Successfully\n" << endl;
            break;
        }
        t_skillNode = t_skillNode->getNextSkillNode();
    }
}
```

Find / read

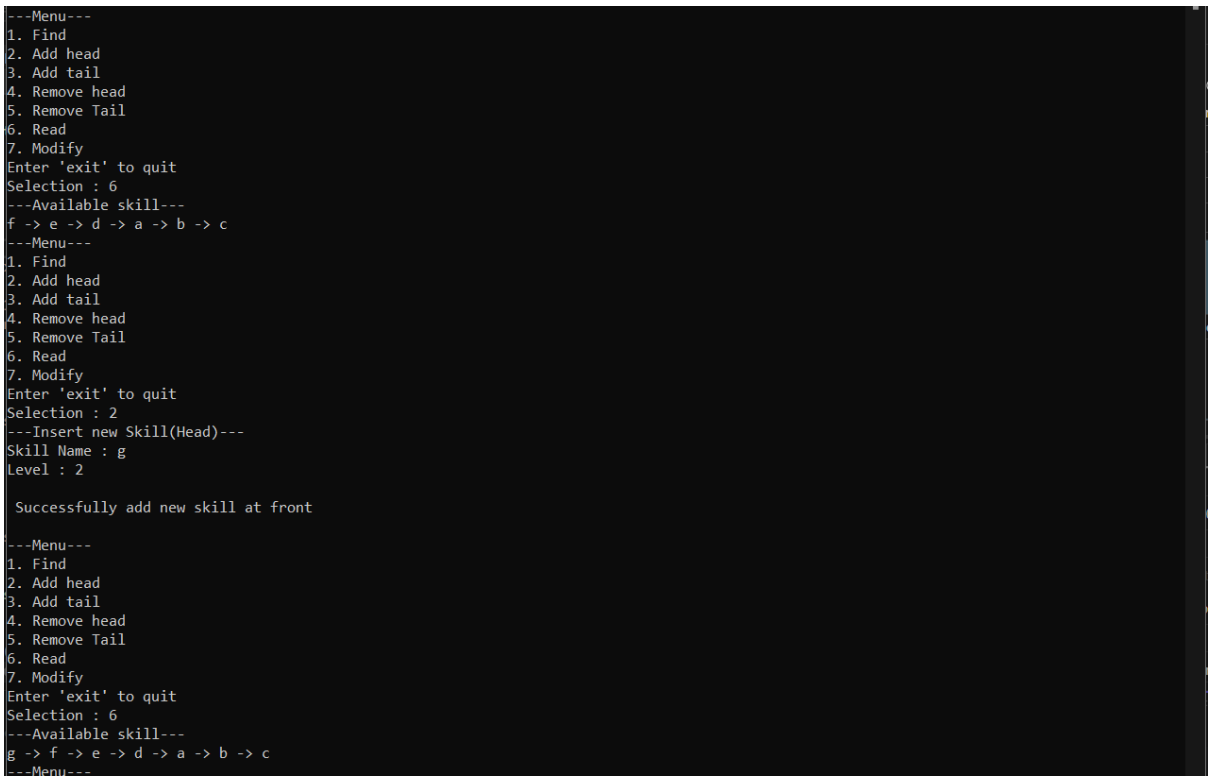
```cpp
void SkillList::find(string s_name) { // find node by search key
    SkillNode* t_skillNode = *&head; // create pointer refrence of head
    bool x = true; // check whether it found or not, found == false, not found == true
    while (t_skillNode->getName() != "") {
        if (t_skillNode->getName() == s_name) {
            t_skillNode->printDetail(); // print node detail
            cout << "Skill Found \n" << endl;
            x = false; // set flag to false
            break; // stop while loop and quit
        }
        if (t_skillNode->getNextSkillNode() == nullptr) { // if next node is null
            break; // break while loop
        }
        t_skillNode = t_skillNode->getNextSkillNode(); // go to next skill node
    }
    if (x) { // if flag is true , print message
        cout << "Skill Not Found \n" << endl;
    }
}

void SkillList::read() {
    SkillNode* t_skillNode = *&head;
    while (t_skillNode->getName() != "") {
        cout << t_skillNode->getName();
        if (t_skillNode->getNextSkillNode() == nullptr) {
            cout << "\n";
            break;
        }
        cout << " -> ";
        t_skillNode = t_skillNode->getNextSkillNode();
    }
}
```

Print from tail/head

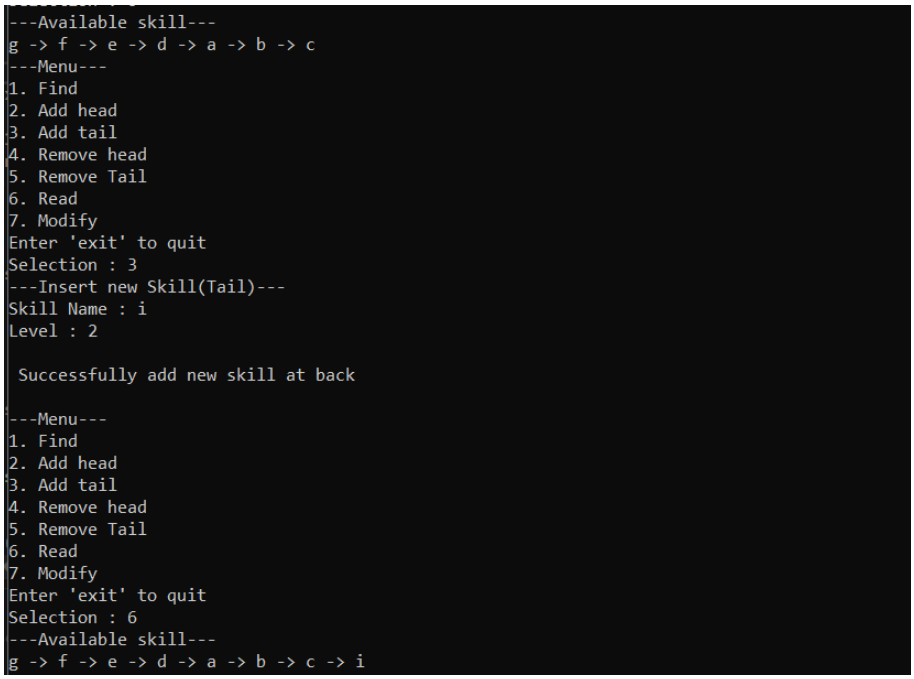
```cpp
void SkillList::PrintFromTail() {
    SkillNode* temp_tail = *&tail;
    int i = 1;
    while (temp_tail->getName() != "") {
        cout << i << ". " << temp_tail->getName() << endl;
        if (temp_tail->getPrevSkillNode() == nullptr) {
            cout << "\n";
            break;
        }
        i++;
        temp_tail = temp_tail->getPrevSkillNode();
    }
}

void SkillList::PrintFromHead() {
    SkillNode* temp_head = *&head;
    int i = 1;
    while (temp_head->getName() != "") {
        cout << i << ". " << temp_head->getName() << endl;
        if (temp_head->getNextSkillNode() == nullptr) {
            cout << "\n";
            break;
        }
        i++;
        temp_head = temp_head->getNextSkillNode();
    }
}
```

Output

Insert front node



Insert back node



Remove head

```
Selection : 6
---Available skill---
g -> f -> e -> d -> a -> b -> c -> i
---Menu---
1. Find
2. Add head
3. Add tail
4. Remove head
5. Remove Tail
6. Read
7. Modify
Enter 'exit' to quit
Selection : 4
f -> e -> d -> a -> b -> c -> i
---Menu---
```

Remove tail

```
f -> e -> d -> a -> b -> c -> i
---Menu---
1. Find
2. Add head
3. Add tail
4. Remove head
5. Remove Tail
6. Read
7. Modify
Enter 'exit' to quit
Selection : 5
f -> e -> d -> a -> b -> c
---Menu---
```

Read

```
---Menu---
1. Find
2. Add head
3. Add tail
4. Remove head
5. Remove Tail
6. Read
7. Modify
Enter 'exit' to quit
Selection : 6
---Available skill---
g -> f -> e -> d -> a -> b -> c
```

Print from tail/head

```
---Menu---
1. Find
2. Add head
3. Add tail
4. Remove head
5. Remove Tail
6. Read
7. Modify
Enter 'exit' to quit
Selection : exit
f -> e -> d -> a -> b -> c
c -> b -> a -> d -> e -> f
```

Modify

```
f -> e -> d -> a -> b -> c
---Menu---
1. Find
2. Add head
3. Add tail
4. Remove head
5. Remove Tail
6. Read
7. Modify
Enter 'exit' to quit
Selection : 7
---Available skill---
1. f
2. e
3. d
4. a
5. b
6. c

Enter Skill Name to Amend : a
Enter New Skill Name : asd
Enter New Level : 2
Amend Data Successfully

---Menu---
1. Find
2. Add head
3. Add tail
4. Remove head
5. Remove Tail
6. Read
7. Modify
Enter 'exit' to quit
Selection : 6
---Available skill---
f -> e -> d -> asd -> b -> c
```

Find

```
---Menu---
1. Find
2. Add head
3. Add tail
4. Remove head
5. Remove Tail
6. Read
7. Modify
Enter 'exit' to quit
Selection : 1
---Find skill---
Enter skill Name : asd
Skill Detail
Name : asd
Level : 2

Skill Found
```