

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	3
Implementation	3
Task 2	4
Implementation	4
Task 3	6
Implementation	6
Task 4	9
Implementation	9
Output.....	10
Task 5	13
Implementation	13
Output.....	17

Task 1

Implementation

```
Task_1 Character
1  #pragma once
2  #include <iostream>
3  #include <string>
4
5  using namespace std;
6
7  class Character {
8  private:
9      string Name; // store the name of player
10     int MaxHp; // store the max hp of player , current hp cnt more than max hp
11     int CurrentHp; // store current hp of player
12 public:
13     Character(string t_name,int t_hp); // constructor of character class
14     int getMaxHp(); // return max hp of player
15     int getCurrentHp(); // return current hp of player
16     string getName(); // return name of the player
17     void setCurrentHp(int dmg); // calculate current hp of player by passing income dmg
18 };
```

```
Task_1 Character
1  #include "Character.h"
2
3  Character::Character(string t_name,int t_hp) {
4      Name = t_name;
5      MaxHp = t_hp;
6      CurrentHp = t_hp;
7  }
8
9  int Character::getMaxHp() {
10     return MaxHp;
11 }
12
13 int Character::getCurrentHp() {
14     return CurrentHp;
15 }
16
17 string Character::getName() {
18     return Name;
19 }
20
21 void Character::setCurrentHp(int dmg) {
22     CurrentHp -= dmg;
23 }
```

Task 2

Implementation

skillNode.h

```
1  #pragma once
2  #include <iostream>
3  #include <string>
4
5  using namespace std;
6
7  class SkillNode {
8  private:
9      string Name; // skill name
10     int Level; // skill level
11     SkillNode* next; // next node
12     SkillNode* prev; // previous skill node
13 public:
14     SkillNode(); // constructor of skill node
15     SkillNode(string t_name, int t_level); // constructor of skill node with parameter
16     int getLevel(); // return level
17     string getName(); // return name
18     SkillNode* getNextSkillNode(); // return next skill Node
19     SkillNode* getPrevSkillNode(); // return prev skill Node
20     void setNextSkillNode(SkillNode* temp); // set next skill node
21     void setPrevSkillNode(SkillNode* temp); // set prev skill node
22     void printDetail(); // print node detail
23     void setLevel(int temp); // set node level
24     void setName(string temp); // set node name
25 };
```

SkillNode.cpp

```
Task_1 SkillNode getPrevSkillNode
1  #include "SkillNode.h"
2
3  SkillNode::SkillNode() {
4      Name = "";
5      Level = 0;
6      next = NULL;
7      prev = NULL;
8  }
9
10
11  SkillNode::SkillNode(string t_name, int t_level) {
12      Name = t_name;
13      Level = t_level;
14      next = NULL;
15      prev = NULL;
16  }
17
18  int SkillNode::getLevel() {
19      return Level;
20  }
21
22  string SkillNode::getName() {
23      return Name;
24  }
25
26  void SkillNode::setLevel(int temp) {
27      Level = temp;
28  }
29
30  void SkillNode::setName(string temp) {
31      Name = temp;
32  }
33
```

```

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

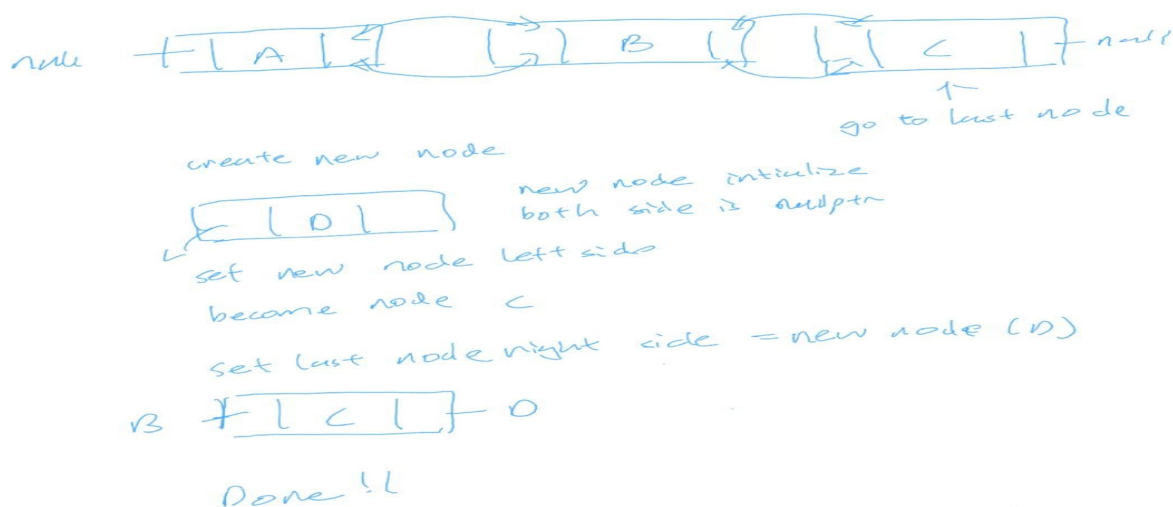
```
main.cpp SkillNode.h SkillNode.cpp Character.h Character.cpp
Task_1 Character
PrintAllSkill()

1 #pragma once
2 #include <iostream>
3 #include <string>
4 #include "SkillNode.h"
5
6
7 using namespace std;
8
9 class Character {
10 private:
11     string Name; // store the name of player
12     int MaxHp; // store the max hp of player , current hp cnt more than max hp
13     int CurrentHp; // store current hp of player
14     //task 3
15     SkillNode* skillNode; //pointer of skill Node
16 public:
17     Character(string t_name,int t_hp,SkillNode* s); // constructor of character class
18     int getMaxHp(); // return max hp of player
19     int getCurrentHp(); // return current hp of player
20     string getName(); // return name of the player
21     void setCurrentHp(int dmg); // calculate current hp of player by passing income dmg
22     //task 3
23     void find(string s_name); //task 3 - find skill exist or not
24     void add(string s_temp, int s_level); // task 3 - add new skill node
25     void remove(string s_name); // task 3 - remove skill node
26     void read(); // task 3 - read the content of skill node
27     void modify(string temp,string s_name, int s_level); // task 3 - modify the content of skill node
28     void PrintAllSkill(); //
29 };
```

Character.cpp

Add

```
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 to 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;
}
```

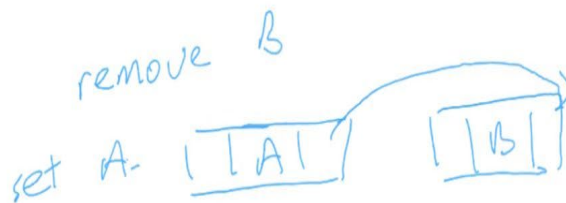
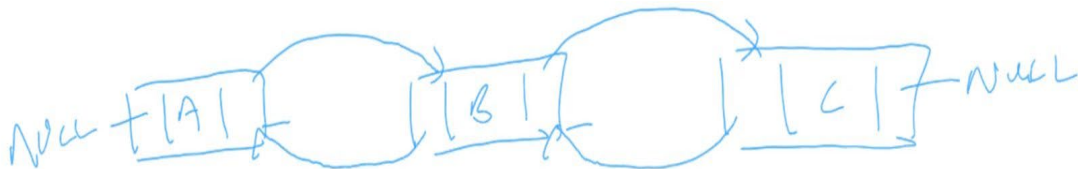


Remove

```

41
42 void Character::remove(string s_name) {
43     SkillNode* t_skillNode = *skillNode; // create pointer reference 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         t_skillNode = t_skillNode->getNextSkillNode(); // go to next node, if not found same skill name
69     }
70 }
71
72

```



Modify

```
2 void Character::modify(string temp, string s_name, int s_level)
3 void Character::modify(string temp, string s_name, int s_level) { // modify node, parameter -> search key, new name, new level
4 SkillNode* t_skillNode = *gskillNode; // copy pointer reference of skill node
5 while (t_skillNode->getName() != "") {
6     if (t_skillNode->getName() == temp) { // if same name found
7         t_skillNode->setName(s_name); // set name
8         t_skillNode->setLevel(s_level); // set level
9         cout << "Amend Data Successfully\n" << endl;
10        break;
11    }
12    t_skillNode = t_skillNode->getNextSkillNode();
13 }
14 }
15
16 void Character::find(string s_name) {
```

Find

```
85
86
87 void Character::find(string s_name) { // find node by search key
88 SkillNode* t_skillNode = *gskillNode; // create pointer reference of skill node
89 bool x = true; // check whether it found or not, found == false, not found == true
90 while (t_skillNode->getName() != "") {
91     if (t_skillNode->getName() == s_name) {
92         t_skillNode->printDetail(); // print node detail
93         cout << "Skill Found \n" << endl;
94         x = false; // set flag to false
95         break; // stop while loop and quit
96     }
97     if (t_skillNode->getNextSkillNode() == nullptr) { // if next node is null
98         break; // break while loop
99     }
100    t_skillNode = t_skillNode->getNextSkillNode(); // go to next skill node
101 }
102 if (x) { // if flag is true, print message
103     cout << "Skill Not Found \n" << endl;
104 }
105 }
106
```

Read

```
void Character::read() {
    SkillNode* t_skillNode = *gskillNode;
    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

```
119
120 void Character::PrintAllSkill() {
121 SkillNode* t_skillNode = *gskillNode;
122 int i = 1;
123 while (t_skillNode->getName() != "") {
124     cout << i << ". " << t_skillNode->getName() << endl;
125     if (t_skillNode->getNextSkillNode() == nullptr) {
126         cout << "\n";
127         break;
128     }
129     i++;
130     t_skillNode = t_skillNode->getNextSkillNode();
131 }
132 }
```


Task 4

Implementation

Main.cpp

```
main.cpp  SkillNode.h  Character.h  Character.cpp  Character.h
Task_1 (Global Scope) Menu()

1  #include "SkillNode.h";
2  #include "Character.h"
3
4  string Menu() { // function of menu
5      string selection;
6      cout << "----Menu----" << endl;
7      cout << "1. Find" << endl;
8      cout << "2. Add" << endl;
9      cout << "3. Remove" << endl;
10     cout << "4. Read" << endl;
11     cout << "5. Modify" << endl;
12     cout << "Enter 'exit' to quit" << endl;
13     cout << "Selection : "; getline(cin, selection);
14     return selection;
15 }
16
17
18 int main() {
19     Character *c;
20     SkillNode* s;
21     string a, b, e, d, selection;
22     cout << "Enter Name : "; getline(cin, a);
23     cout << "Enter MaxHP : "; getline(cin, b);
24
25     s = new SkillNode();
26     c = new Character(a, stoi(b), s);
27     c->add("a", 1);
28     c->add("b", 2);
29     c->add("c", 3);
30     while (selection != "exit") { // if selection != next, loop keep going
31         selection = Menu(); // call function menu
32         if (selection == "1") { // if 1 == find
33             cout << "----Find skill----" << endl;
34             cout << "Enter skill Name : "; getline(cin, e); // enter skill name
35             c->find(e); // call find in character class and pass search key
36         }
37         else if (selection == "2") { // if 2 == add
38             cout << "----Insert new Skill----" << endl;
39             cout << "Skill Name : "; getline(cin, e); // enter skill name
40             cout << "Level : "; getline(cin, d); // enter skill level
41             c->add(e, stoi(d)); // call add in character class and pass name and level by parameter
42         }
43         else if (selection == "3") { // if 3 == remove
44             cout << "----Remove skill----" << endl;
45             c->PrintAllSkill(); // call print skill function in character class to print current skill
46             cout << "Enter skill Name : "; getline(cin, e); // enter skill name wish to remove
47             c->remove(e); // call remove in character class and pass search key(name) by parameter
48         }
49         else if (selection == "4") { // if 4 == read
50             cout << "----Available Skill----" << endl;
51             c->read(); // read in character class to print current link list
52         }
53         else if (selection == "5") { // if 5 == modify
54             cout << "----Available skill----" << endl;
55             c->PrintAllSkill(); // call print all skill in character class
56             cout << "Enter Skill Name to Amend : "; getline(cin, b); // enter skill name wish to amend
57             cout << "Enter New Skill Name : "; getline(cin, d); // enter new skill name
58             cout << "Enter New Level : "; getline(cin, e); // enter new level
59             c->modify(b, d, stoi(e));
60             // call modify function in character class and pass, b d e , search key/new name/new level by parameter
61         }
62     }
63     return 0;
64 }
65
```

Output

Find

```
---Menu---
1. Find
2. Add
3. Remove
4. Read
5. Modify
Enter 'exit' to quit
Selection : 1
---Find skill---
Enter skill Name : a
Skill Detail
Name : a
Level : 1
Skill Found
```

Read and Add

```
---Menu---
1. Find
2. Add
3. Remove
4. Read
5. Modify
Enter 'exit' to quit
Selection : 4
---Available Skill---
a -> b -> c -> d
---Menu---
1. Find
2. Add
3. Remove
4. Read
5. Modify
Enter 'exit' to quit
Selection : 2
---Insert new Skill---
Skill Name : e
Level : 6

    Successfully create new skill

---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 : █
```

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

```
k5 (Global Scope) main()
1  #include "Character.h"
2  #include "SkillList.h"
3
4  string Menu() { // function of menu
5      string selection;
6      cout << "----Menu----" << endl;
7      cout << "1. Find" << endl;
8      cout << "2. Add head" << endl;
9      cout << "3. Add tail" << endl;
10     cout << "4. Remove head" << endl;
11     cout << "5. Remove Tail" << endl;
12     cout << "6. Read" << endl;
13     cout << "7. Modify" << endl;
14     cout << "Enter 'exit' to quit" << endl;
15     cout << "Selection : "; getline(cin, selection);
16     return selection;
17 }
18
19
20 int main() {
21     Character* c;
22     SkillNode* s;
23     string a, b, e, d, selection;
24     cout << "Enter Name : "; getline(cin, a);
25     cout << "Enter MaxHP : "; getline(cin, b);
26
27     s = new SkillNode();
28     c = new Character(a, stoi(b));
29     c->addback("a", 1);
30     c->addback("b", 2);
31     c->addback("c", 3);
32     c->addfront("d", 3);
33     c->addfront("e", 3);
34     c->addfront("f", 3);
35     while (selection != "exit") { // if selection != next, loop keep going
36         selection = Menu(); // call function menu
37         if (selection == "1") { // if 1 == find
38             cout << "----Find skill----" << endl;
39             cout << "Enter skill Name : "; getline(cin, e); // enter skill name
40             c->find(e); // call find in character class and pass search key
41         }
42         else if (selection == "2") { // if 2 == add front
43             cout << "----Insert new Skill(Head)----" << endl;
44             cout << "Skill Name : "; getline(cin, e); // enter skill name
45             cout << "Level : "; getline(cin, d); // enter skill level
46             c->addfront(e, stoi(d)); // call addfront in character class and pass name and level by parameter
47         }
48         else if (selection == "3") { // if 3 == add back
49             cout << "----Insert new Skill(Tail)----" << endl;
50             cout << "Skill Name : "; getline(cin, e); // enter skill name
51             cout << "Level : "; getline(cin, d); // enter skill level
52             c->addback(e, stoi(d)); // call addback in character class and pass name and level by parameter
53         }
54         else if (selection == "4") { // if 4 == remove front
55             c->removefront(); // remove first node
56             c->read();
57         }
58         else if (selection == "5") {
59             c->removeback(); // remove last node
60             c->read();
61         }
62         else if (selection == "6") {
63             cout << "----Available skill----" << endl;
64             c->read();
65         }
66         else if (selection == "7") {
67             cout << "----Available skill----" << endl;
68             c->PrintFromHead(); // print skill from head
69             cout << "Enter Skill Name to Amend : "; getline(cin, b); // enter skill name wish to amend
70             cout << "Enter New Skill Name : "; getline(cin, d); // enter new skill name
71             cout << "Enter New Level : "; getline(cin, e); // enter new level
72             c->modify(b, d, stoi(e));
73             // call modify function in character class and pass, b d e , search key/new name/new level by parameter
74         }
75         else { // when exit will print existing node in this 2 format
76             c->PrintFromHead();
77             c->PrintFromTail();
78         }
79     }
80     return 0;
81 }
```

Character.h

```
1  #pragma once
2  #include <iostream>
3  #include <string>
4  #include "SkillList.h"
5
6  using namespace std;
7
8  class Character {
9  private:
10     string Name; // store the name of player
11     int MaxHp; // store the max hp of player , current hp cnt more than max hp
12     int CurrentHp; // store current hp of player
13     SkillList SL;
14 public:
15     Character(string t_name, int t_hp); // constructor of character class
16     int getMaxHp(); // return max hp of player
17     int getCurrentHp(); // return current hp of player
18     string getName(); // return name of the player
19     void setCurrentHp(int dmg); // calculate current hp of player by passing income dmg
20     //task 3
21     void find(string s_name); //task 3 - find skill exist or not
22     void addfront(string s_temp, int s_level); // task 3 - insert node at front
23     void addback(string s_temp, int s_level); // task 3 - insert node at back
24     void removefront(); // task 3 - remove skill node
25     void removeback(); // task 3 - remove skill node
26     void read(); // task 3 - read the content of skill node
27     void modify(string temp, string s_name, int s_level); // task 3 - modify the content of skill node
28     void PrintFromTail();
29     void PrintFromHead();
30 };
31
```

Character.cpp

```
1 void Character::addfront(string s_temp, int s_level) { // call skilllist add front
2     SL.addfront(s_temp,s_level);
3 }
4
5 void Character::addback(string s_temp, int s_level) { // call skilllist add back function
6     SL.addback(s_temp, s_level);
7 }
8
9 void Character::removefront() { // call skill list remove front function
10     SL.removefront();
11 }
12
13 void Character::removeback() { //call skill list remove back function
14     SL.removeback();
15 }
16
17 void Character::modify(string temp, string s_name, int s_level) { // call skill list modify function
18     SL.modify(temp, s_name, s_level);
19 }
20
21 void Character::find(string s_name) { // call skill list find function
22     SL.find(s_name);
23 }
```

```
24
25 void Character::read() { // call skill list read function
26     SL.read();
27 }
28
29 void Character::PrintFromTail() { // call skill list read from tail function
30     SL.PrintFromTail();
31 }
32
33 void Character::PrintFromHead() { // call skill list read from head function
34     SL.PrintFromHead();
35 }
36
```

SkillList.h

```
1  #pragma once
2  #include <iostream>
3  #include <string>
4  #include "SkillNode.h"
5
6  using namespace std;
7
8  class SkillList {
9  private:
10     SkillNode* skillNode;
11     SkillNode* head;
12     SkillNode* tail;
13 public:
14     SkillList();
15     void find(string s_name); //task 5 - find skill exist or not
16     void addfront(string s_temp, int s_level); // task 5 - add new skill node at first
17     void addback(string s_temp, int s_level); // task 5 - add new skill node at last
18     void removefront(); // task 5 - remove frsrit skill node
19     void removeback(); // task 5 - remove last skill node
20     void read(); // task 5 - read the content of skill node
21     void modify(string temp, string s_name, int s_level); // task 3 - modify the content of skill node
22     void PrintFromTail(); //
23     void PrintFromHead();
24 };
```

SkillList.cpp

Add back

```
#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 reference of skill node
        tail = *skillNode; // become pointer reference 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

```
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 reference of skill node
        tail = *skillNode; // become pointer reference 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

```
54 void SkillList::removeback() { // when more that 2 node
55     SkillNode* current_tail = *tail; // get current tail
56     if (current_tail->getPrevSkillNode() != nullptr) { // when more that 2 node
57         SkillNode* temp = current_tail->getPrevSkillNode(); // save current tail prev node to temp
58         temp->setNextSkillNode(nullptr); // set temp next node to null
59         tail = temp; // temp become tail
60     }
61     else { // only 1 node
62         skillNode = nullptr;
63         head = nullptr;
64         tail = nullptr;
65     }
66 }
67
68 void SkillList::modify(string temp, string s_name, int s_level) { // modify node, parameter -> search key, new name, new level
69     SkillNode* t_skillNode = *skillNode; // copy pointer reference
70     while (t_skillNode->getName() != "") {
71         if (t_skillNode->getName() == temp) { // if same name found
72             t_skillNode->setName(s_name); // set name
73             t_skillNode->setLevel(s_level); // set level
74             cout << "Amend Data Successfully\n" << endl;
75             break;
76         }
77         t_skillNode = t_skillNode->getNextSkillNode();
78     }
79 }
80
81
82
```

Find / read

```
83 void SkillList::find(string s_name) { // find node by search key
84     SkillNode* t_skillNode = *head; // create pointer reference of head
85     bool x = true; // check whether it found or not, found == false, not found == true
86     while (t_skillNode->getName() != "") {
87         if (t_skillNode->getName() == s_name) {
88             t_skillNode->printDetail(); // print node detail
89             cout << "Skill Found \n" << endl;
90             x = false; // set flag to false
91             break; // stop while loop and quit
92         }
93         if (t_skillNode->getNextSkillNode() == nullptr) { // if next node is null
94             break; // break while loop
95         }
96         t_skillNode = t_skillNode->getNextSkillNode(); // go to next skill node
97     }
98     if (x) { // if flag is true , print message
99         cout << "Skill Not Found \n" << endl;
100     }
101 }
102
103 void SkillList::read() {
104     SkillNode* t_skillNode = *head;
105     while (t_skillNode->getName() != "") {
106         cout << t_skillNode->getName();
107         if (t_skillNode->getNextSkillNode() == nullptr) {
108             cout << "\n";
109             break;
110         }
111         cout << " -> ";
112         t_skillNode = t_skillNode->getNextSkillNode();
113     }
114 }
```

Print from tail/head

```
115 void SkillList::PrintFromTail() {
116     SkillNode* temp_tail = *tail;
117     int i = 1;
118     while (temp_tail->getName() != "") {
119         cout << i << ". " << temp_tail->getName() << endl;
120         if (temp_tail->getPrevSkillNode() == nullptr) {
121             cout << "\n";
122             break;
123         }
124         i++;
125         temp_tail = temp_tail->getPrevSkillNode();
126     }
127 }
128
129 void SkillList::PrintFromHead() {
130     SkillNode* temp_head = *head;
131     int i = 1;
132     while (temp_head->getName() != "") {
133         cout << i << ". " << temp_head->getName() << endl;
134         if (temp_head->getNextSkillNode() == nullptr) {
135             cout << "\n";
136             break;
137         }
138         i++;
139         temp_head = temp_head->getNextSkillNode();
140     }
141 }
142
143
```


Output

Insert front node

```
--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 -> 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 : 2
---Insert new Skill(Head)---
Skill Name : g
Level : 2

    Successfully add new skill at front

--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
--Menu--
```

Insert back node

```
---Available skill---
g -> 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 : 3
---Insert new Skill(Tail)---
Skill Name : i
Level : 2

    Successfully add new skill at back

--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 -> i
```

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

```
Selection : 4
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
Menu
```

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
```