Assessment Cover Sheet

This Assessment Cover Sheet is only to be attached to

ASSESSMENT DETAILS

Unit title	Data Structures and Pattern	Tutorial /Lab Group	1	Office use only
Unit code	COS30008	Due date	14 OCT 2022	
Name of lecturer/tutor	Dr, Mark Tee Kit Tsun			
Assignment title	Problem Set 2			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)		

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)		(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

Description

This task Is going to create a console application allow user to input command and react the appropriate action based on the command user input. Between this , getter and setter is necessary to create to read and write content into attributes and fields created.

Implementation

Entities.h

```
class Entities {
private:
	int fEntityID;
	string fName;
	double fHP;
	double fCurrentHP;
	string fMessage;

public:
	Entities(); // constructor of entities
	~Entities(); // destructor of entity
	int getfEntityID(); // getter of entity id
	void setfEntityID(int temp);// setter of entity id
	string getfName(); // getter of entity name
	void setfName(string temp); // setter of entity name
	string getfMessage();
	double getfCurrentHP();

	friend istream& operator>>(istream& ist,Entities& object);
	//friend ostream& operator<<(ostream& ost,const Entities& object);
};
```

Entities.cpp

```
#include "Entities.h"

Entities::Entities() {
	fName = "";
	fEntityID = 0;
	fHP = 200;
	fCurrentHP = 200;
}

Entities::~Entities() {

}

int Entities::getfEntityID() {
	return fEntityID;
}

string Entities::getfName() {
	return fName;
}

void Entities::setfEntityID(int temp) {
	fEntityID = temp;
}

void Entities::setfName(string temp) {
	fName = temp;
```

```
}

string Entities::getfMessage() {
	return fMessage;
}

double Entities::getfCurrentHP() {
	return fCurrentHP;
}

istream& operator>>(istream& ist, Entities& object) {
	vector<string> splitString; // create an vector to store array
	string temp; // create varaible to store input command temporary
	istringstream iss; // split string into word
	cout << "\nEnter (Exit - quit , increase current HP - HEAL 20) : " <<
endl;
	getline(ist, temp); // read command
	if (temp.compare("EXIT") == 0) { // if input is EXIT
		object.fMessage = temp; // store full command into fmessage
	}
	else {
		iss.str(temp); // split string into word
		for (string s; iss >> s;) { // for loop until the end of the array
in iss
			splitString.push_back(s); // push word splited by iss into
splitString(vector)
		}
		object.fCurrentHP += stod(splitString[1]); // add up current hp and
heal hp
		object.fMessage = temp; // store full command into fmessage
	}

	return ist;
}
```

Main.cpp

```
#pragma once
#include "Entities.h"

int main() {

	Entities e;
	string name;
	cout << "Player Name : "; getline(cin,name);
	e.setfName(name);
	while (e.getfMessage().compare("EXIT")) { // checking user input is exit
or command
		cin >> e; //input by friend operator
		if (e.getfMessage().compare("EXIT") == 0) { // if input is exit
			cout << "\nProblem Set 2 Task 1 Exiting......" << endl;
//showing quiting state
		}
		else {  // if input is command
			cout << "Current HP : " << e.getfCurrentHP() << endl; //
showing current hp
		}
	}

	return 0;
}
```

Output

```
Microsoft Visual Studio Debug Console
Player Name : Alex

Enter (Exit - quit , increase current HP - HEAL 20) :
HEAL 40
Current HP : 240

Enter (Exit - quit , increase current HP - HEAL 20) :
HEAL 60
Current HP : 300

Enter (Exit - quit , increase current HP - HEAL 20) :
EXIT

Problem Set 2 Task 1 Exiting......
Destructor Work

C:\Users\green\Desktop\Swinburne\Data Structure\ProblemSet2\Task\Task1\x64\Debug\Task1.exe (process 21996) exited with c
ode 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

Troubleshooting

Research for the function how to compare between the string. Because in c++ cannot simply compare two string by using operator ==.

Task 2

Description

This task is to familiar with iterator by create two type of array, one dimensional array and two-dimensional array represent two different types of bags, create 2 iterator classes, “Iterator1D” and “Iterator2D” to implements 4 operators, “++”,”--”, “==”, “*”. After all creation done, use all the iterator object created in class “Iterator1D” and “Iterator2D” to do the testing in main function by passing the array created.

Concept

Iterator is a pointer-like object, generic algorithm uses it to get elements from and store elements to various containers without affect others. It provides generic and efficient way to perform algorithm defined that suitable for them.

Implementation

Iterator1D.h

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

using namespace std;

class Iterator1D {
private:
	const string* item;
	const int size;
	int index;
public:
	Iterator1D(const string temp_arr[], const int temp_size);
	~Iterator1D();
	const string& operator*() const;
	Iterator1D& operator++();
	Iterator1D& operator--();
	void operator==(const Iterator1D& temp) const;
};
```

Iterator2D.h

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

using namespace std;

class Iterator2D {
private:
	const string *item;
	const int size;
	int index;
public:
	Iterator2D(const string *temp_item, const int temp_size);
	~Iterator2D();
	const string& operator*() const;
	Iterator2D& operator++();
	Iterator2D& operator--();
	void operator==(const Iterator2D& temp) const;
};
```

Iterator1D.cpp

```
#include "Iterator1D.h"

Iterator1D::Iterator1D(const string temp_arr[], const int
temp_size) :item(temp_arr), size(temp_size) {
	index = 0;
};

Iterator1D::~Iterator1D(){};

const string& Iterator1D::operator*() const { //this deference operator will
return the current positioned item.
	return item[index];
}

Iterator1D& Iterator1D::operator++() { // increment operator will back to the
first index once the index is out of bound
	index++;
	if (index >= size) {
		cout << "Last index in array, cannot increment anymore : ";
		index = size - 1;
	}
	return *this;
}

Iterator1D& Iterator1D::operator--() { // decrement operator will stop decrease
the index once the position of index at 0
	if (index != 0) {
		index--;
	}
	return *this;
}

void Iterator1D::operator==(const Iterator1D& temp) const { //
	bool x = (index == temp.index) && (item == temp.item);
	if (x == 1) {
		cout << "Same item in both array" << endl;
	}
	else {
		cout << "Different item in both array" << endl;
	}
}
```

Iteractor2D.cpp

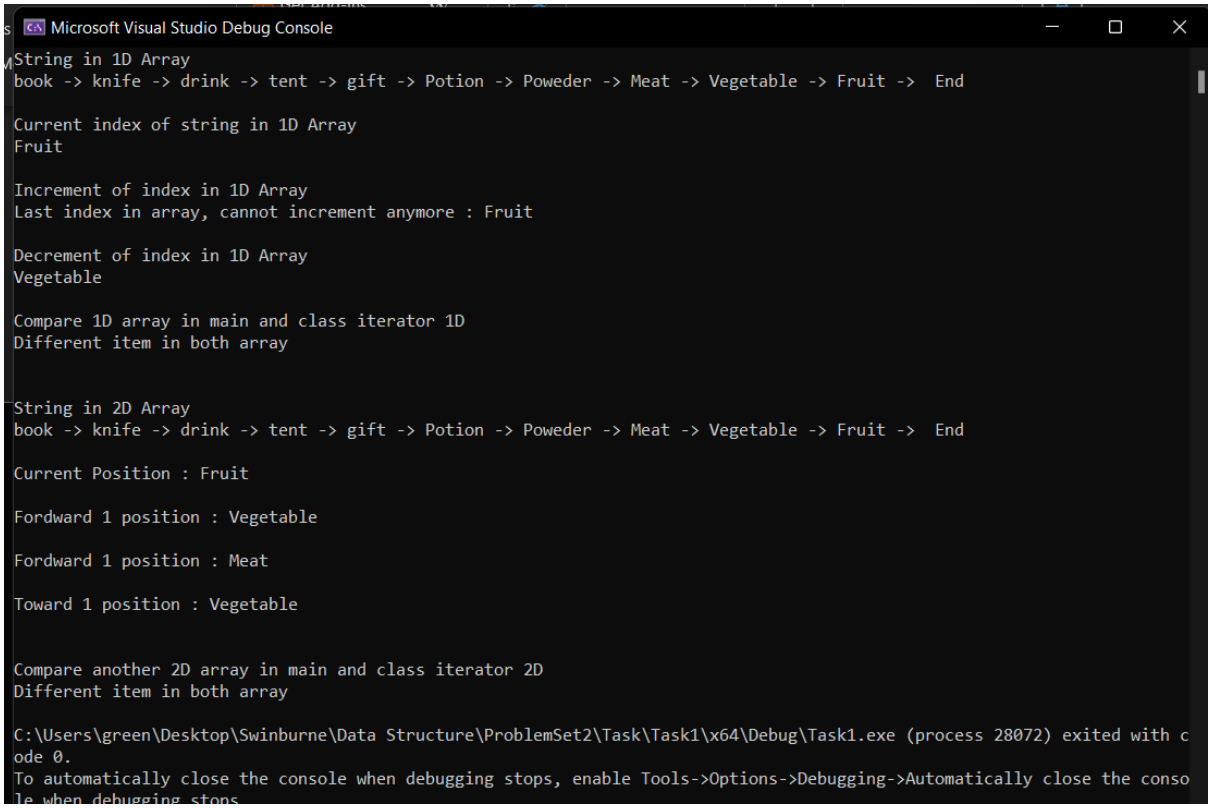
```
#include "Iterator2D.h"

Iterator2D::Iterator2D(const string *temp_item, const int
temp_size) :item(temp_item), size(temp_size) {
	index = 0;
}

Iterator2D::~Iterator2D() {};

const string& Iterator2D::operator*() const { //this deference operator will
return the current positioned item.
	return *(item + index);
}

Iterator2D& Iterator2D::operator++() { // increment operator will back to the
first index once the index is out of bound
	index++;
	if (index >= size) {
		cout << "Last index in array, cannot increment anymore : ";
		index = size - 1;
	}
	return *this;
}

Iterator2D& Iterator2D::operator--() { // decrement operator will stop decrease
the index once the position of index at 0
	if (index != 0) {
		index--;
	}
	return *this;
}

void Iterator2D::operator==(const Iterator2D& temp) const { //
	bool x = (index == temp.index) && (*(item + index) == *(temp.item +
index));
	if (x == 1) {
		cout << "Same item in both array" << endl;
	}
	else {
		cout << "Different item in both array" << endl;
	}
}
```

Main.cpp

```
#pragma once
#include "Iterator1D.h"
#include "Iterator2D.h"

int main() {

	//-----------------------
	//Task 2
	string item[] =
{"book","knife","drink","tent","gift","Potion","Poweder","Meat","Vegetable","Frui
t"};
	string item2[][4] = {
		{"book","knife","drink","tent"},
		{"gift","Potion","Poweder","Meat"},
		{"Vegetable","Fruit"}
	};
	Iterator1D i(item,10);
	Iterator1D i_comp(item, 10);
	Iterator2D i2(*item2, 10);
	Iterator2D i2_comp(*item2,10);

	cout << "String in 1D Array " << endl;
	for (int j = 0; j < 10; j++) {
		if (j == 0) {
			cout << i.operator*() << " -> ";
		}
		else {
			cout << i.operator++().operator*() << " -> ";
		}
	}
	cout << " End" << endl;

	cout << "\nCurrent index of string in 1D Array" << endl;
	cout << i.operator*();

	cout << "\n\nIncrement of index in 1D Array" << endl;
	cout << i.operator++().operator*();

	cout << "\n\nDecrement of index in 1D Array" << endl;
	cout << i.operator--().operator*();

	cout << "\n\nCompare 1D array in main and class iterator 1D" << endl;
	i.operator==(i_comp);


	cout << "\n\nString in 2D Array " << endl;
	for (int j = 0; j < 10 ; j++) {
		if (j == 0) {
			cout << i2.operator*() << " -> ";
		}
		else {
			cout << i2.operator++().operator*() << " -> ";
		}
	}
	cout << " End" << endl;

	cout << "\nCurrent Position : " << i2.operator*() << endl;
	cout << "\nFordward 1 position : " << i2.operator--().operator*() << endl;
	cout << "\nFordward 1 position : " << i2.operator--().operator*() << endl;
	cout << "\nToward 1 position : " << i2.operator++().operator*() << endl;
```

```
        cout << "\n\nCompare another 2D array in main and class iterator 2D" <<
endl;
        i2.operator==(i2_comp);

        return 0;
}
```

Output



Troubleshooting

Research for the pointer using on multiple dimensional arrays, how to get and store. Research for the iterator operator==() uses and how it works.

Task 3

Description

Modify entity class to hold iterator1D and iterator2D class. Add public method/function in the entity class that's "InventoryNext", "InventoryPrev", "InventoryGet" to perform they own action. Use Operator == to make sure no out it mix/max range. After that, allow iterator1D transfer item to iterator2D bag by using 3 function created.

Implementation

Entities.h

```
	void grab(Iterator1D &temp); //store Itertor in entities class
	void grab(Iterator2D &temp); //store Itertor in entities class
	void InventoryNext(Iterator1D temp); // Move Itertor1D to next index
	void InventoryNext(Iterator2D temp);// Move Itertor2D to next index
	void InventoryPrev(Iterator1D& temp); // Move Itertor1D to previous index
	void InventoryPrev(Iterator2D& temp);// Move Itertor2D to previous index
	const string& InventoryGet(int temp); // get current index item
	void ReceiveItem(Entities& temp,Iterator1D temp_1); // player 2 receive
item from player 1
	void ReceiveItem(Entities& temp, Iterator2D temp_1);// player 1 receive
item from player 2
	void RemoveItem(Iterator2D temp_1); // player 2 remove item
	void RemoveItem(Iterator1D temp_1); // player 1 remove item
	void Print(Iterator1D); // print current item iterator1D class have
	void Print(Iterator2D); // print current item iterator2D class have
```

Main.cpp

```
Entities p1;
	Entities p2;
	string name;
	string item[] =
{"book","knife","drink","tent","gift","Potion","Poweder","Meat","Vegetable","Frui
t","empty"};
	string item2[][4] = {
		{"a","b","c","d"},
		{"e","f","g","h"},
		{"i","j","empty"}
	};
	Iterator1D i(item,11);
	Iterator2D i2(*item2, 11);
	cout << "Player 1 Name : "; getline(cin,name);
	p1.setfName(name);
	p1.grab(i);

	cout << "Plyear 2 Name : "; getline(cin,name);
	p2.setfName(name);
	p2.grab(i2);


	string input;
	while (input.compare("exit")) {
		string input_item;
		cout << "\nSelect Role " << endl;
		cout << "1. " << p1.getfName() << " transfer to " << p2.getfName()
<< endl;
		cout << "2. " << p2.getfName() << " trasnfer to " << p1.getfName()
<< endl;
		cout << "Select : "; getline(cin,input);
		while (input_item.compare("5")) {
```

```
                    cout << "\nSelect item transfer " << endl;
                    cout << "1. Previous " << endl;
                    cout << "2. Next " << endl;
                    cout << "3. Transfer " << endl;
                    cout << "4. Check bag" << endl;
                    cout << "5. exit " << endl;
                    if (input.compare("1") == 0) {
                          cout << "Current Item : " << p1.InventoryGet(1) <<
endl;;
                          cout << "Enter : ";  getline(cin, input_item);
                          if (input_item.compare("1") == 0) {
                                p1.InventoryPrev(i);
                          }
                          else if (input_item.compare("2") == 0) {
                                p1.InventoryNext(i);
                          }
                          else if (input_item.compare("3") == 0) {
                                p2.ReceiveItem(p1,i2);
                                p1.RemoveItem(i);
                          }
                          else if (input_item.compare("4") == 0) {
                                cout << "\nIterator 1D bag (Alex): ";
p1.Print(i);
                                cout << "Iterator 2D bag (ben): "; p2.Print(i2);
                          }
                    }
                    else if(input.compare("2") == 0) {
                          cout << "Current Item : " << p2.InventoryGet(2) <<
endl;
                          cout << "Enter : ";  getline(cin, input_item);
                          if (input_item.compare("1") == 0) {
                                p2.InventoryPrev(i2);
                          }
                          else if (input_item.compare("2") == 0) {
                                p2.InventoryNext(i2);
                          }
                          else if (input_item.compare("3") == 0) {
                                p1.ReceiveItem(p2,i);
                                p2.RemoveItem(i2);
                          }
                          else if (input_item.compare("4") == 0) {
                                cout << "\nIterator 1D bag (Alex): ";
p1.Print(i);
                                cout << "Iterator 2D bag (ben): "; p2.Print(i2);
                          }
                    }
              }
      }
```

Entities.cpp

```
void Entities::grab(Iterator1D &temp) { // set iterator1D item
      i1 = &temp;
}
void Entities::grab(Iterator2D &temp) {
      i2 = &temp;
} // set iterator2D item

void Entities::InventoryNext(Iterator1D temp) {
      if (temp.operator==(i1->operator++()) == 0) { // if max of the index, stop
moving and move backward 1 time to make sure always in the last index
```

```
		i1->operator--();
	}

}

void Entities::InventoryNext(Iterator2D temp) {
	if (temp.operator==(i2->operator++()) == 0) { // if max of the index, stop
moving and move backward 1 time to make sure always in the last index
		i2->operator--();
	}
}

void Entities::InventoryPrev(Iterator1D& temp) {
	if (temp.operator==(i1->operator--()) == 0) { // // if min of the index,
stop moving and move toward 1 time to make sure always in the first index
		i1->operator++();
	}
}

void Entities::InventoryPrev(Iterator2D& temp) {
	if (temp.operator==(i2->operator--()) == 0) { // // if min of the index,
stop moving and move toward 1 time to make sure always in the first index
		i2->operator++();
	}
}

const string& Entities::InventoryGet(int temp) { // get current item of the index
in Iterator class, 1 is iterator1D otherwise is Iterator 2D
	if (temp == 1) {
		return i1->operator*();
	}
	else {
		return i2->operator*();
	}
}


void Entities::ReceiveItem(Entities &temp, Iterator1D temp_1) {
	i1->initialize();
	while (temp_1.operator==(*i1)) {
		if (i1->operator*() == "empty") {
			i1->setItem(temp.InventoryGet(2));
			break;
		}
		i1->operator++();
		if (temp_1.operator==(*i1) == 0) {
			i1->operator--();
			break;
		}
	}
}
void Entities::ReceiveItem(Entities& temp, Iterator2D temp_1) {
	i2->initialize();
	while (temp_1.operator==(*i2)) {
		if (i2->operator*() == "empty") {
			i2->setItem(temp.InventoryGet(1));
			break;
		}
		i2->operator++();
		if (temp_1.operator==(*i2) == 0) {
			i2->operator--();
			break;
		}
```

```
	}
}


void Entities::RemoveItem(Iterator2D temp_1) { // set transfered item slot to
empty
	i2->setItem("empty");
}

void Entities::RemoveItem(Iterator1D temp_1) {// set transfered item slot to
empty
	i1->setItem("empty");
}

void Entities::Print(Iterator2D temp) { // print content of iterator2D class
	i2->initialize();
	while (temp.operator==(*i2)) {
		cout << i2->operator*();
		i2->operator++();
		if (temp.operator==(*i2) == 0) {
			i2->operator--();
			cout << endl;
			break;
		}
		else {
			cout << "->";
		}
	}
}

void Entities::Print(Iterator1D temp) {// print content of iterator1D class
	i1->initialize();
	while (temp.operator==(*i1)) {
		cout << i1->operator*();
		i1->operator++();
		if (temp.operator==(*i1) == 0) {
			i1->operator--();
			cout << endl;
			break;
		}
		else {
			cout << "->";
		}
	}
}
```

Output

```
Player 1 Name : alex
Plyear 2 Name : ben

Select Role
1. alex transfer to ben
2. ben trasnfer to alex
Select : 1

Select item transfer
1. Previous
2. Next
3. Transfer
4. Check bag
5. exit
Current Item : book
Enter : 2

Select item transfer
1. Previous
2. Next
3. Transfer
4. Check bag
5. exit
Current Item : knife
Enter : 3

Select item transfer
1. Previous
2. Next
3. Transfer
4. Check bag
5. exit
Current Item : empty
Enter : 4

Iterator 1D bag (Alex): book->empty->drink->tent->gift->Potion->Poweder->Meat->Vegetable->Fruit->empty
Iterator 2D bag (ben): a->b->c->d->e->f->g->h->i->j->knife

Select item transfer
1. Previous
2. Next
3. Transfer
4. Check bag
5. exit
Current Item : empty
Enter :
```

```
Select item transfer
1. Previous
2. Next
3. Transfer
4. Check bag
5. exit
Current Item : empty
Enter : 5

Select Role
1. alex transfer to ben
2. ben trasnfer to alex
Select : 2

Select item transfer
1. Previous
2. Next
3. Transfer
4. Check bag
5. exit
Current Item : knife
Enter : 1

Select item transfer
1. Previous
2. Next
3. Transfer
4. Check bag
5. exit
Current Item : j
Enter : 1

Select item transfer
1. Previous
2. Next
3. Transfer
4. Check bag
5. exit
Current Item : i
Enter : 3
```

```
Select item transfer
1. Previous
2. Next
3. Transfer
4. Check bag
5. exit
Current Item : empty
Enter : 4

Iterator 1D bag (Alex): book->i->drink->tent->gift->Potion->Poweder->Meat->Vegetable->Fruit->empty
Iterator 2D bag (ben): a->b->c->d->e->f->g->h->empty->j->knife

Select item transfer
1. Previous
2. Next
3. Transfer
4. Check bag
5. exit
Current Item : knife
Enter :
```

Troubleshooting

No problem encountered.