Seneca | SCHOOL OF INFORMATION AND COMMUNICATIONS TECHNOLOGY

TERM	COURSE NAME	COURSE CODE	VERSION
Winter 2019	Object-Oriented Software Development using C++	OOP345	A

Name	(write your full name here)
Student Number	(write your student number here)
Section	(write your section number here)

DATE: April 18 2019

TIME ALLOWED: 2 hours
PERCENTAGE: 20%
Part A Concept Questions 10
Part B Short Coding Questions 20
Part C Walkthroughs 10
Part D Word Problem 30
Total Marks: 70
PROFESSOR(S): Asam Gulaid, Jimmy Or, Fardad Soleimanloo, Chris Szalwinski

SPECIAL INSTRUCTIONS:

1. A double-sided printed letter-size reference sheet (8.5 x 11 or A4) is permitted. The sheet must include your name and student number and must be submitted with the exam.
2. Write your answers legibly in the spaces provided
3. No electronic or messaging devices permitted

This exam includes a *cover page*, plus nine (9) pages of questions.

SENECA'S ACADEMIC HONESTY POLICY
As a Seneca student, you must conduct yourself in an honest and trustworthy manner in all aspects of your academic career. A dishonest attempt to obtain an academic advantage is considered an offense, and will not be tolerated by the College.

APPROVED BY:

Suzanne Abraham, Interim Chair, School of ICT

Part A: Concept Questions (10 marks)

Answer each question in the space provided.

1 – Name one example of a Standard Template Library (STL) container that is optimized for adding and removing elements from

a) at the front and the back of a sequence

b) at random location

2 – What does the acronym RAII stand for? Name an STL class that implements RAII in C++?

3 – The algorithm category of the STL consists of three distinct libraries. Name the three libraries and describe the functionality of each one in a short phrase

a)

b)

c)

4 – Describe the difference between a C function pointer and a C++ functor with reference to state

5 – Describe the role of a provider in multi-threading. Name an STL class template for a provider.

Part B: Short Coding Questions (20 marks)

6 – Fill in the blanks (10 out of 20 marks)

```
// Fill in 5 blanks correctly for full marks – 6 blanks correctly for 12
marks
#include <iostream>
#include <iomanip12>
#include <vector>

#include _____________________________
using namespace std;

const int k = 10;

int main() {

	vector<int> vector(5, 13);

	vector[3] = k;

	// find the first element that is equal to k

	auto it = find_if(

		_____________________________________________,

		_____________________________________________,

		_____________________________________________________________________
	);

	cout << setw(5) << _________________ // value of the element

		<< " found at vector["

		<< ____________________________________ // index of the element

		<< "]" << endl;
}
```

Output:

```
10 found at index [3]
```

Reference:

InputIterator std::find_if(InputIterator f, InputIterator l, Fn predicate)

7 – Correct the Errors (10 out of 20 marks)

```
// 5 errors with corrections for full marks – 6 correct answers for 12 marks

#include <iostream>
using namespace std;

struct A {

	const int x; // do not change

public:

	A(const int c)
	{
		x = c;
	}

	void operator()() // do not change
	{
		cout << x << '/n';
	}
};

void foo(void* c) // do not change
{
	cout << *c << ':';
}

int main(int argc, char* argv[])
{
	int c{ 5 };	// do not change

	foo(&c);	// do not change

	A a(c);		// do not change

	(c++)++;

	if (argv[1][0] == 'x' && argc > 1)

		a;

	return 0;
}
```

Output:

5:5

Part C: Walkthroughs (10 marks)

8 – What is the output of the following program (5 marks)?

```
#include <iostream>
#include <vector>
#include <utility>
using namespace std;

class Page {
	const char* title;
public:
	Page(const char* t) : title(t) {}
	void display() const {
		cout << title << endl;
	}
};
ostream& operator<<(ostream& os, const Page& p) {
	p.display();
	return os;
}

int main() {
	vector<Page> site;
	site.push_back(Page("One"));
	site.push_back(Page("Two"));
	for (auto& it : site)
		cout << it;
	cout << "-----" << endl;

	site.push_back(Page("Three"));
	vector<Page> newSite{ move(site) };
	for (auto& it : newSite)
		cout << it;
	cout << "-----" << endl;

	site.push_back(Page("Four"));
	for (auto& it : site)
		cout << it;
	cout << "-----" << endl;
}
```

9 – What is the output of the following program (5 marks)?

```
#include <iostream>
#include <thread>
using namespace std;

thread_local int c{ 4 };


void task(int* a) {
	*a += c++;
}


int main() {
	int a[]{ 6, 7, 8 };
	thread t1(task, &a[0]);
	thread t2(task, &a[1]);
	t1.join();
	t2.join();
	task(&a[2]);
	for (int& e: a)
		cout << e << endl;
}
```

Part D:– 8 – Word Problem – Grocery Cart (30 marks)

In namespace **sict** design and code a class named **Grocery** for holding information about a single grocery item. The information includes a description of the item, its price and its taxable status. A **Grocery** item cannot be copied, assigned in any way, but it can be moved to a new item. Your class includes the following public member functions:

A three-argument constructor that receives an unmodifiable reference to a **string** containing the description of the item, a **double** holding the price of the item and an unmodifiable reference to a **string** holding the taxable status of the item. Admissible tax strings begin with **H** (for HST) and **P** (for PST). Your constructor does not accept tax strings that begin with any other character and reports an exception as shown on the first line of the output below.

A move constructor

A query named **display** that receives a reference to an **ostream** object. This query inserts item information into the **ostream** object as shown in the example below. The description is left-justified in a field width of **FWDescription** and the price is right-justified in a field width of **FWPrice**, both of which are defined outside the translation unit for your class as shown in the listing below.

The **main** function that uses your implementation calls a global function named **loadCart**, which loads the grocery information from a file named **userFile** into the shopping cart named **cart**:

```
#include <iostream>
#include <iomanip>
#include <vector>
#include "Grocery.h"
using namespace std;
using namespace sict;

int FWPrice{ 5 };
int FWDescription{ 10 };
void loadCart(const char* userFile, vector<Grocery>& cart);

int main(int argc, char* argv[]) {
	cout << setprecision(2) << fixed;
	vector<Grocery> cart;
	loadCart(argv[1], cart);
	cout << "Items in Cart:" << endl;
	for (auto& e : cart) {
		e.display(cout);
	}
}
```

For the **userFile** listed on the left, this **main** function displays the information listed on the right:

apples 1.29 pears 1.49 milk 5.00	*unlisted tax symbol* : potatoes 2.49 GST Items in cart:

gum 1.49 HST insurance 19.99 PST potatoes 2.49 GST yogurt 3.49	apples 1.29 pears 1.49 milk 5.00 gum 1.49 HST insurance 19.99 PST yogurt 3.49

8 (a) – Grocery.h (5 out of 30 marks)

For full marks include all the statements necessary for successful compilation.

8 (b) – Grocery.cpp (15 out of 30 marks)

For full marks include all the statements necessary for successful compilation.

8 (c) – loadCart.cpp (10 out of 30 marks)

Complete the missing code for this function:

```
// TODO: header files




using namespace std;

void loadCart(const char* filename, vector<Grocery>& cart) {
	ifstream file(filename);
	if (!file) exit(1);
	while (file) {
		string description;
		double price;
		string tax = " ";
		file >> description >> price;
		bool isTaxed = file.get() != '\n';
		if (isTaxed)
			file >> tax;

		// TODO: create the item, move it to the cart and handle the
exception









	} // end while
```

```
}
```