Quiz 8 (Last one. You did it!) **(Fall 2019)**

Section: B

Name: Nicholas De Franco

Marks: 1/ /10

Stud #: 106732183

Please Answer in complete sentences:

1) What library do we include to allow access to the threading class? **(1)**

The library we include to allow access to the threading class is <thread>. ✓

2) What is a **race condition** in terms of multi threading? **(3)**

A race condition is when 2 or more threads attempt to update the same memory location and the same time. It can also be when two processes attempt to write to the same file at the same time. ✓

3) What is a **deadlock** in terms of multi threading? **(3)**

A deadlock occurs when threads are waiting on each each other to finish execution. This will cause the threads to be blocked indefinately. ✓

4) What is a **process** and what is a **thread**? **(3)**

A process is an instance of a program executed on a host platform.

A thread is a sequence of instrucion that are executed independantly by the OS's scheduler. They are light-weight processes that belong to a host process.

Bonus Question: What in your opinion was the **hardest topic/concept** in oop345 this semester? If you could **change 1 thing** in the course what would it be? **(1)**

- The hardest topic this semester in my opinion was this week's topic. I never understood multithreading when I tried to learn it on my own time before coming to Seneca.
- If I were to change 1 thing in this course I would be reducing the amount

Quiz 7 (Second last one. You can do it!) **(Fall 2019)**

Section: B

Name: Nicholas DeFranco

Marks: 11 /10

Stud #: 106732183

Please Answer in complete sentences:

1) If a raw pointer is a built-in type that holds an address in memory, and a smart pointer wraps raw pointers: **what type of pointer have we mostly been using in this class? (2)**

The pointer we have been mostly using in this class is a raw pointer.

2) Explain the dangers of a **raw pointer** with dynamic memory **going out of scope? (3)**

It is the programmers responsibility to manage memory pointed to by a raw pointer

A raw pointer that points to dynamic memory (a resource) must be deallocated before the pointer goes out of scope. If the pointer goes out of scope we lose the address of the dynamic memory causing a memory leak.

3) Explain the dangers of a **unique** *smart pointer* **going out of scope? (3)**

The dangers of a unique smart pointer involves the fact that if a raw pointer was pointing to the same address, that raw pointer will not be set to nullptr when the resource/object is deallocated by the unique smart pointer instead it is just left as wild pointer without any indication.

4) What library are **unique_ptr** and **shared_ptr** a part of? Hint: (**std:: is not the answer I'm looking for)(2)**

The std::unique_ptr<> class template and the std::shared_ptr<> class template are defined in the <memory> header file.

Bonus Question: What Stream object did we suggest using in the last Workshop?(1)

Stringstream was suggested to be used in the last workshop.

Quiz 6 **(Fall 2019)**

Name: Nicholas Defranco

Stud #: 106732183

Section: B

Marks: 9 /10

1) Which **file stream class** allows for both write and read access to a file? **(3)**

fstream allows for both write and read access to a file. ✓

2) If I have an **ostream** object called **fout**, how can I check that there is a successful connection to the file I want to write to? **(3)**

To check if the object successfully connected to the file you can either call the is_open() member function or call the overload ! operator right after attempting to open the file. ✓ ✓

3) Look at the code below, what is a major difference between variable **refWrap** and **refPoint**? How does calling them differ? **(2)**

```
int a = 5;

std::reference_wrapper<int> refWrap = a;

int* refPoint = &a;
```

modifications to refwrap directly modify a.

refwrap object stores a reference to the integer variable a. The refwrap object can directly access the value of a without the need of dereferencing an address. A reference stored in a reference_wrapper is essentially a pointer that is already dereferenced which simplies syntax. ✓

refPoint stores the address of a. In order to access the value of a the address must be followed (dereferenced). This can sometimes cause confusing syntax.

Bonus Question: In your opinion, what was the hardest part of the midterm and why? **(1)**

The hardest part of the midterm was the programming section. This is because it is sometimes difficult to remember syntax without the compiler not being able to test code.

Quiz 5 (Fall 2019)

Section: B

Name: Nicholas Defranco 11

Marks: /10

Stud #: 106732183

1) What is an iterator in terms of containers? (1)

An iterator provides a way to traverse a data structure that is not stored contiguosly in memory. They are used to simulate sequential access. |

2) Lists have sub-optimal storage. Why is that so, when compared to vectors and deques? (2)

Lists have sub-optimal storage as they are not optimised for fast access. Lists must iterate sequential through the elements every time in order to retrieve data. unlike vectors and deques which can use the subscripting operator for faster access. 2

3) What is a Deque? How does it manage its storage? (2)

A Deque is a doubly ended queue data structure. The data structure is optimised for insertions and deletions at either end of the data structure. It is stored in dynamic memory (free store) and can change its size as needed. 2

4) What is a vector? How does it differ from other containers? (2)

A vector is a dynamic array data structure. The data structure is optimised for insertions and deletions only at the end of data structure. unlike the other containers this container is guaranteed to be stored contiguosly in memory. 2

5) When talking about **Stacks** vs **Queues**, how does each one retrieve and remove its data? (3)

Stack is a data structure that works in FILO context. Elements at the top of the stack (last entered) are removed first. 3

Queue is a data structure that operates in FIFO context, Elements entered first exit first.

Bonus Question: What happens if I ran this code? int * a; delete[] a; (1)

(fast top) run-time error as you are de-allocating memory that does not belong ... (you don't know ✓

Quiz 4 (Fall 2019)   Section: B

Name: Nicholas Defranco   Stud #: 10673215 3   Marks: 11 /10

1) For this lambda expression: [ ] ( ) { } Tell me what each pair of brackets do? **(3)** 3

[ ] ↑ capture list identifies access specification for non-local variables

( ) ↑ parameter list

{ } ↑ contains the body of the lambda expression (definition)

2) What **typing** do you need when assigning a lambda to a variable? Why? **(2)** 2

auto& ← we always access functions indirectly. Any time we call a function we transfer control to its address in memory. Therefore, if we wish to store the lambda in a variable we must make it of function pointer type or a reference (auto& simplifies syntax)

3) What does a recursive function **need/require** to finish the code? What happens if a recursive function **runs too many times**? **(2)** 2

A recursive function requires an exit condition to terminate the function.

If a recursive function runs too many times it will cause a stack overflow.

4) In exception handling what **2 parts** do we need when reporting and handling errors? What do they do? **(3)** 3

exception reporting

throw expression ← is of previously defined type.

reports an exception to be (hopefully) caught by an catch block

handling errors

try block → contains code that could throw an exception.

catch block(s) → contains code that handles a particular exception based on the type specified in the parenthesis after the catch keyword.

Bonus Question: If I have x = 5, and y = !(!x); what is the value of x? What is the value of y? **(1)**

$y = !(!5)$

$y = !(0)$

$\boxed{y = 1}$

$\boxed{x = 5}$

✓

Quiz 3 **(Fall 2019)**

Section: B

Name: Nicholas Defranco

Marks: 11 /10

Stud #: 106732183

Instructions: I am looking for **1 – 2 sentence** answers, **not** full paragraphs **20 min** total.

1) Explain the creation/destruction of objects with an **association**. Ex. An object called **Player** is **associated with a Team (3)** 3

An association is a relationship where one class does not own the other. There are independant of each other. This means the relating class manages its construction and destruction on their own. construction and destruction of the related type occurs outside of the class

2) Explain the creation/destruction of objects with **aggregation**. Ex. An object called **School may have** objects of **Students**. **(2)** 2

Aggregation is a composition where a class does not manage the construction and destruction of the aggregatee type. construction and destruction of the aggregatee type occurs outside of the class.

3) Explain the creation/destruction of objects with **composition**. Ex. An object called **buildings** has-a **has/is composed of** an object called **rooms. (2)** 2 sub-object destroyed before base object

Composition involves complete ownership of an object. This means the composer class is responsible for the construction and destruction of its component before it destroys itself

4) If we have a variable **int i = 5,** one of these operations work and one doesn't. Which one works and why doesn't the other one work? **(3)** 3 B will not work as i++ returns a rvalue when the operand should be an lvalue to work

A. ++(++i) **B. (i++)++**

A will work as the pre-fix increment operator requires an lvalue. Pre-fix incremement returns an l value allowing the statment to work

Bonus Question: How can you swap the values of two ints without a third variable or functions? **(1)**

int a = 5;
int b = 6;

a = b - a; ✓
b = b - a;
a = b + a;

Quiz 2 (Fall 2019)

Section: B

Name: Nicholas DeFranco

Marks: 11 /10

Stud #: 106732183

Instructions: This is a simple quiz; you have **15 min** to write it. I am looking for **1 – 2 sentence** answers, **not** full paragraphs.

1) What is the difference between a **Concrete type** and an **Abstract type**?(2) 2 type

concrete type - complete n that can be instantiated as ✓ the definition is known. used to ~~provide~~ give detail that abstract classes do not

Abstract type - incomplete type that is missing details, cannot be instantiated as definition is unknown.

provides an interface that classes ~~similar~~ similar functions to ~~[illegible]~~

2) What 2 things do you need to make a **pure virtual** function and describe what each does?(4) 4

✓

Virtual

virtual keyword allows for dynamic dispatch

= 0

Identifies function as pure (no definition will be associated with this specific declaration)

3) What are the **2 admissible types** (types that allow for substitution) for a template's parameters. E.g. template <????> ... (2) 2

2 admissible types that allow for substitution are ✓ type template perameters (typename and class) and non-type template perameters (ex, integer or ~~[illegible]~~ enum type)

4) How can we specify a default value for a template? (2) 2

Similar syntax to default value for function parameters. Set parameter identifier equal to a value.

ex: template < typename T = long, int size = 5

Bonus Question: What was the main cause of the WS1 submitter problem? (1) 1

one directory in the path to the files was missing the execute permission ✓ for the users

Quiz 1 **(Fall 2019)**

Section: B

Name: Nicholas Defrance

Marks: 10 /10

Stud #: 106732193

Instructions: This is a simple quiz; you have **15 min** to write it. I am looking for **1 – 2 sentence** answers, **not** full paragraphs.

1) Simply Explain the difference between a **Signed** and **Unsigned** variable. How does their range capacity differ (what max/min values can they hold)? **(4)** 4

Signed variables can store negative values, 0, and positive values. Unsigned variables can store 0 and positive values.

signed $-(2^{(n-1)}) \leftrightarrow 2^{(n-1)}-1$

unsigned $0 - (2^{n}-1)$

n is number of bits

2) What is the difference between a **Lvalue** and a **Rvalue**? Or when would you use them? **(2)** 1

Lvalue is a value that can be used as an operand in an expression which an Rvalue cannot

*: both an Lvalue and Rvalue can be assigned to a variable

$a = a + b$, — Lvalue

$a = var{+}{+}$ ((var+1)++) — Rvalue — illegal

3) Why would you use a **move** copy constructor or a **move** assignment operator instead of the regular copy constructor or assignment operator? **(3)** 3

if an object is about to go out of scope (leaving a function) and be returned instead of copying the returned result we can move the contents for efficiency

4) How do we specify variable inference when creating a new variable. What 2 parts do we need? **(1)** 1

auto i = 7; ✓

① keyword ② initial value

Bonus Question: What does the = operator return? (in most cases) (1)

A reference to the left operand ✓