**CPSC 1020 Spring 2019**

**EXAM #2**

**This is the written portion of Exam 2.**

**This is a Closed Book exam. Please keep your notes and your computers closed. You have 50 minutes to complete the Exam.**

**Question 1(9 Points)C++ Memory Allocation, Structs, Returning memory to OS**

struct Pixel

{

unsigned char red, green, blue;

};

**Part 1:**

Assume Row and Column have been defined. In C++, dynamically allocate the memory for a **2D Array** of type Pixel called **pixArray.**

**Part 2**

In C++, write the code needed to give the memory allocated in part one back to the OS.

**Question 2 (3 Points) shared\_ptrs**

**Consider the following program. What is the output?**

#include<iostream>

#include<memory>

using namespace std;

OUTPUT:

int main()

{

int\* s = new int;

shared\_ptr<int> s1(s);

shared\_ptr<int> s2 = s1;

shared\_ptr<int> s3(s2);

cout << "s1 count " << s1.use\_count() << endl;

cout << "s2 count " << s2.use\_count() << endl;

cout << "s3 count " << s3.use\_count() << endl;

s = nullptr;

return 0;

}

**Question 3 (6 Points) 2D vector**

Part 1: Create an empty 2 Dimensional vector of integers called **vec**.

(You did this in a lab and it is only one line of code)

Part 2: Assume row and col have been defined. Resize **vec** to be of size row x col.

(Again, you did this in lab and can be done in only one line of code. )

**Question 4: (3 Points) Reference variables**

What is the output of this program:

#include <iostream>

using namespace std;

int main()

{

OutPut:

double first = 15.3;

double second = 18.5;

double& ref\_d = first;

cout << first << endl;

cout << ref\_d << endl;

ref\_d = second;

cout << first << endl;

cout << ref\_d << endl;

second = 19.8;

cout << first << endl;

cout << ref\_d << endl;

return 0;

}

**Question 5 (10 Points total) Classes**

**For the following two parts, consider the class on Attachment 1:**

**Part 1**

Write the code for the following operator=:

MyClass MyClass::operator=(const MyClass& origClass) {

cout << "operator = " << endl;

}

**Part 2**

What is the output of this program? See attachment 1

**For the remaining questions circle the correct choice to question.**

**Each of the multiple choice questions are 3 points, T/F are 2 points**

**Question 6**

Assuming that arr is an array identifier, the statement sum += \*arr;

A) is illegal in C++.

B) will always result in a compiler error.

C) adds the value stored in arr[0] to sum.

D) adds the address of the pointer arr to sum.

E) None of the these

**Question 7**

True / False:

Memory cannot be allocated after a program is already running.

**Question 8**

The delete operator should only be used on pointers that

A) have not yet been used.

B) have been correctly initialized.

C) point to storage allocated by the new operator.

D) are appropriately dereferenced.

E) None of the these

**Question 9**

True / False:

A pointer with the value 0 (zero) is called the NULL pointer.

**Question 10**

Which of the following statements correctly deletes a dynamically-allocated array pointed to by p?

A) delete p;

B) p delete[ ];

C) delete [ ] p;

D) delete array p;

E) None of the these

**Question 11**

Suppose that a function dynamically allocates a block of memory with a local pointer variable p pointing to the allocated block. Suppose further that there are no other pointers referencing that block of memory, and the function returns without doing a delete on p. Then

A) the pointer p becomes a dangling pointer.

B) the compiler will automatically deallocate the memory pointed to by p.

C) the program will suffer from memory leaks.

D) the returning function will throw the bad\_alloc exception.

E) None of the these

**Question 12**

The set of operations **supported** by the unique\_ptr class include

A) the dereferencing operators \* and ->, the post and pre increment operators ++, and the post and pre decrement operators --.

B) the dereferencing operators \* and ->.

C) the assignment operator.

D) the delete operator.

E) None of the these

**Question 13**

A pointer variable is designed to store

A) any legal C++ value.

B) only floating-point values.

C) a memory address.

D) an integer.

E) None of the these

**Question 14**

True / False:

C++ does not perform array bounds checking.

**Question 15**

The statement cout << \*ptr; will output

A) the value stored in the variable whose address is contained in ptr.

B) the string "\*ptr".

C) the address of the variable stored in ptr.

D) the address of the variable whose address is stored in ptr.

E) None of the these

**Question 16**

The statement cin >> \*p;

A) stores the keyboard input into the variable p.

B) stores the keyboard input into the pointer called p.

C) is illegal in C++.

D) stores the keyboard input into the variable pointed to by p.

E) None of the these

**Question 17**

Objects that work like pointers but have the ability to automatically delete dynamically allocated memory are \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

1. Destructor
2. Smart Pointer
3. Constructor
4. None of the these

**Question 18**

When three different objects of a class are created, they are said to be separate \_\_\_\_\_\_\_\_ of the class.

A) members

B) ADTs

C) instances

D) children

E) None of these

**Question 19**

When the body of a member function is defined inside a class declaration, it is called a(n) \_\_\_\_\_\_\_\_ function.

A) static

B) global

C) inline

D) conditional

E) constructor

**Question 20**

A constructor may have a return type of

A) int

B) bool

C) void

D) any of these.

E) none of these.

**Question 21**

A destructor is a member function that

A) Is used to remove old unneeded objects.

B) causes the program to end.

C) Is automatically called when an object is destroyed.

D) can only be called by the main function of a program.

E) None of these.

**Question 22**

A(n) \_\_\_\_\_\_\_\_ member function may be called by a statement in a function that is outside of the class.

A) inline

B) public

C) private

D) declared

E) constructor

**Question 23**

Accessors are sometimes called \_\_\_\_\_\_\_\_ functions and mutators are sometimes called \_\_\_\_\_\_\_\_ functions.

A) set, get

B) get, set

C) public, private

D) private, public

E) regular, inline

**Question 24**

If Circle is the name of a class, which of the following statements would create a Circle object named myCircle?

A) myCircle Circle;

B) myCircle Circle();

C) Circle myCircle;

D) Circle myCircle();

E) None of these

**Question 25**

The \_\_\_\_\_\_\_\_ is used to protect important data.

A) default constructor

B) class protection operator

C) public access specifier

D) private access specifier

**Question 26**

A class may have \_\_\_\_\_\_\_\_ default constructor(s) and \_\_\_\_\_\_\_\_ destructor(s).

A) only one, only one

B) only one, more than one

C) more than one, only one

D) more than one, more than one

E) no, only one

**Question 27**

True / False:

The range-based for loop may be used with arrays, but not with vectors.

**Question 28**

True / False:

Arrays can be passed to functions, but individual array elements cannot be.

**Question 29**

True / False:

In C++ If you attempt to store more data in an array than it can hold, the compiler will issue a compile time error.

**Question 30**

If employee is an array of objects with a public member function named setHoursWorked, which of the following statements correctly calls that function for the employee object in array element 5?

A) employee.setHoursWorked[5] = 40;

B) employee[5].setHoursWorked = 40;

C) employee.setHoursWorked[5](40);

D) employee[5].setHoursWorked(40);

E) setHoursWorked(employee[5], 40);

*/\*Attachment 1\*/*

#include <iostream>

using namespace std;

class MyClass

{

private:

int\* dataObj;

public:

MyClass();

MyClass(const MyClass& origClass);

~MyClass();

MyClass operator=(const MyClass& origClass);

void SetDataObj(const int setVal);

int GetDataObj() const;

};

*// Default constructor*

MyClass::MyClass() {

cout << "Default" << endl;

*//you may assume this function has been implemented.*

}

*// Copy constructor*

MyClass::MyClass(const MyClass& origClass) {

cout << "Copy" << endl;

*//you may assume this function has been implemented.*

}

*// Destructor*

MyClass::~MyClass() {

cout << "Destructor" << endl;

if (dataObj != 0) {

delete dataObj;

}

}

*/\*You will implement this function\*/*

*//Assignment operator*

MyClass MyClass::operator=(const MyClass& origClass) {

cout << "operator = " << endl;

}

int MyClass::GetDataObj() const {

return \*dataObj;

}

void MyClass::SetDataObj(const int setVal) {

\*dataObj = setVal;

}

void SomeFunction(MyClass localObj) {

*// Do something with localObj*

}

int main() {

MyClass tempClassObj; *// Create object of type MyClass*

MyClass anotherClassObj;

*//make sure you understand what is called here*

anotherClassObj = tempClassObj;

*// Calls SomeFunction(), hint: tempClassObj is passed by value*

SomeFunction(tempClassObj);

return 0;

}