**CPSC 1020 Fall18**

**EXAM #2**

**This is an individual exam. You may not use, books, phones, or any other outside resources. This exam consist of a total of 35 points, however, your grade will be graded on % basis. As an example, if you get 28 points correct your grade will be 28/35 which means you earned a grade of 80%**

**Question 1 (1 Points) Class relationships**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ models a “has-a” relationship between classes.

**Question 2 (3 Points) Vector**

In class we discussed three advantages vectors have over arrays. In one sentence each describe each advantage.



**Question 3 (1 Point) Vector**

True / False

Pop\_back() is a function provided by the vector class that returns the value of the last element of a vector.

**Question 4 (1 Point) Vector**

When we access arrays and vectors using the bracket operator [ ], C++ does not provide bounds checking. However, the vector class provides a function that is safer to use and provides bounds checking.

Rewrite the following code using the function provided by the vector class, that provides bounds checking.

vector<int>test(5);

test[3] = 10;

**Question 5 (2 Points) Vector Functions**

push\_back and resize are functions provided by the vector class. In no more than 2 sentences. explain what each of these code snippets do.

Code Snippet 2:

vector<int> scores(15);

scores.resize(25);

Code Snippet 1:

vector<int>test;

test.push\_back(75);

**Question 6 (2 Points) C++ Memory Allocation**

Below are two common problems when managing dynamically allocated memory.

A \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ occurs if no-longer-needed dynamic memory is not given back to the operating system.

What is the solution to this problem:

A pointer is \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ if it contains the address of memory that has been freed.

What is the solution to this problem:

**Question 7 (5 Points) C++ Memory Allocation**

Rewrite the following **C** code in **C++**

int \*\*arr = (int \*\*)malloc(row \* sizeof(int \*));

for (i=0; i<row; i++)

{

arr[i] = (int \*)malloc(col \* sizeof(int));

}

for(i = 0; i < n; i++)

{

free(ptr[i]);

}

free(ptr);

**Question 8 (2 Points) Data Hiding**

We discussed two reasons why information hiding is a good concept. Name one:

**Question 9 (2 Points) Inline function**

We discussed two ways to create an inline function. In no more than two sentences each, describe the two ways to create an inline function.

a.

b.

**Question 10 (1 Point) Inline function**

With respect to an inline function:

What happens during compile time if a function is inline.

**Question 11 (2 points) Basic Classes**

In no more than 2 sentences, explain the Rule of Three.

**Question 12 (2 Points) Reference**

We discussed three major differences between references and pointers. In no more than 2 sentences each discuss 2 of them.

a.

b.

**For the following question use Attachment 1**

**Question 13 (1 Point) Reference**

What is the output of the program. Put your output in the box below.

Output:

**Question 14 (1 Point) Copy Constructor**

C++ will provide a default copy constructor for each class. Under what circumstances should we implement our own copy constructor rather than rely on the default copy constructor?

**For the following 2 questions you will use Attachment 2.**

**Question 15 (3 Points) operator =**

Write the code for the following operator=:

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

cout << "operator = " << endl;

}

**Question 16 (6 Points) Functions calls**

What is the output of the this program?

*/\*Attachment 1\*/*

#include <iostream>

using namespace std;

int main()

{

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;

}

*/\*Attachment 2\*/*

#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;

}