**CPSC 1020 SUMMER 2017**

**EXAM #2**

**There is a total of 106 points. (6 points extra credit) Good Luck!!**

**[General classes, copy constructors, static variables, inline functions, operator overloading, destructor]**

**Question 1:**

**This covers the next several questions. (48 points total)**

**Write the implementation for the Array class and answer any questions pertaining to this class. You will not implement all of the class, just the functions indicated.**

class Array

{

private:

int size;

int \*ptr;

/\*This static variable is to determine how many instances of

\*the class Array is instantiated (created)\*/

static int arrayCount;

public:

Array();

Array(int aSize);

Array(const Array &obj);

~Array();

int getSize()const;

static int getArrayCount()

{

return arrayCount;

}

Array &operator=(const Array &obj);

void print(int) const;

void setElement(int e, int value)

{

ptr[e] = value;

}

int &Array::operator[](int sub) const;

};

**9 points**

The function setElement(int e, int value) is considered an inline function. Explain why it is considered an inline function?

The compiler does something special when it sees an inline function, what is it?

What is an advantage of an inline function?

**6 points Implement this constructor**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*The regular constructor**

**\* Remember every time a regular constructor gets called an**

**\* instance of the class is instantiated (created)**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

Array::Array(int size)

{

}

**6 points**

**Implement this constructor**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*The copy constructor**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

Array::Array(const Array &obj)

{

}

**3 points**

**Implement the destructor**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*The destructor gives the allocated memory**

**\*back to the OS also decrements static variable**

**\*remember when a destructor is called an instance of**

**\*the class is destroyed**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

Array::~Array()

{

}

**3 points**

**Implement the getter**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*The getter for size**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

int Array::getSize() const

{

}

**6 points**

**Implement the operator=**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*The overloaded = operator for assignment**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

Array &Array::operator=(const Array &rhs)

{

}

**3 points**

**Implement the print function**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*prints the value of an element of the array. The**

**\*element to be printed is determined by the**

**\* subscript passed into the function**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

void Array::print(int subscript) const

{

}

**6 points**

**Implement the overloaded [ ] operator**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* Overloaded [] operator. If the subscript is out of**

**\* bounds print an error message and exit.**

**\* The argument is a subscript.**

**\* This function returns a reference to the element**

**\* in the array indexed by the subscript.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

int &Array::operator[](int sub) const

{

}

**6 points**

**Describe where the static variable should be initialized and write the code to initialize the static variable.**

**[General Object Oriented Programming]**

**Question 2: (3 points)**

Fill in the blank.

When a class “has a” instance of another class, this is an example of

**[Member Initialization List]**

**Question 3: (6 points)**

**Consider the code below and complete the Person constructor using the member initialization list technique.**

class Address{

private:

int house;

string street;

string city;

string state;

int zip;

public:

Address(int h, string str, string c, string st, int z);

//setter

//getter

};

class Date{

private:

int month;

int day;

int year;

public:

Date();

Date(int m, int d, int y);

//getters

//setters

};

class Person{

private:

string last;

string first;

string email;

Address address;

Date bday;

public:

Person();

Person(string l, string f, string e, int house, string street, string city, string state, int zip,

int month, int day, int year);

//setters and getters

};

Person::Person()

Person::Person(string l, string f, string e, int number, string street,string city, string state, int zip, int month, int day, int year)

**[Inheritance]**

**Question 4: (3 points)**

What type of relationship is realized by inheritance? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**[Protected data access]**

**Question 5: (2 points) True / False**

Protected members are accessible to member functions as well as member functions of all derived classes.

**[Inherited access specification]**

**Question 6: (12 points)**

**Consider the following class:**

class Base

{

private:

int a;

protected:

int b;

int c;

void setA(int x) {a = x;}

public:

void setB(int y) {b = y;}

void setC(int z) {c = z;}

};

**Answer the following questions (there will be two parts to this question)**

1. Suppose another class *Derived*, is derived from the *Base* class. Here is the first line of its declaration:

class Derived : private Base

Determine whether each member of the Base class is private, protected, public, or inaccessible:

a \_\_\_\_\_\_\_\_\_\_\_\_\_

b \_\_\_\_\_\_\_\_\_\_\_\_\_

c \_\_\_\_\_\_\_\_\_\_\_\_\_

1. Suppose another class *Derived*, is derived from the *Base* class. Here is the first line of its declaration:

class Derived : protected Base

Determine whether each member of the *Base* class is private, protected, public, or inaccessible:

a \_\_\_\_\_\_\_\_\_\_\_\_\_

b \_\_\_\_\_\_\_\_\_\_\_\_\_

c \_\_\_\_\_\_\_\_\_\_\_\_\_

**[Constructors]**

**Question 7: (6 points)**

**Consider the following class definition:**

class NumberArray{

private:

double \*aPtr;

int arraySize;

public:

NumberArray operator=(const NumberArray &right);

NumberArray(const NumberArray &);

NumberArray(int size, double value);

~NumberArray() { if (arraySize > 0) delete [] aPtr; }

void print();

void setValue(double value);

};

Given the following main.cpp, answer the questions listed beside each line in main.cpp.

/\*You can assume the appropriate #includes are here.\*/

int main()

{

NumberArray first(3, 10.5); **What constructor is called**:

NumberArray second = first; **What constructor is called**:

NumberArray third(5, 12.4); **What constructor is called**:

third = first;

**The above line of code will cause the operator = to be called. This line of code will cause a constructor to be called as well as the operator = overloaded function.**

**What constructor will be called and why?**

return 0;

}

**[Abstract and static functions]**

**Question 8: (6 points)**

Consider the implementation of the following static function:

static void Person::printInfo( )

{

cout << numPersons;

}

In the process of writing the code for the Person class I would like to call the printInfo() function. However, Person is an abstract class, when I create an instance of Person in order to call the printInfo() function I get a compile error.

Why am I getting an error and **explain** how to call printInfo() without an instance of the class. Also, give an example of how to call the printInfo function.

**[Vectors]**

**Question 9: (1 point each; total of 4)**

Given the following:

vector<int> myList(4);

myList.at(0) = 10;

myList.at(1) = 85;

myList.at(2) = 63;

How many elements does the vector declaration create?

1. 0
2. 1
3. 3
4. 4

What value is assigned into myList.at(1)?

1. 1
2. 10
3. 85
4. 0

What value does int curr = myList.as(2) assign to curr?

1. 63
2. 85
3. 2
4. invalid index

What are the contents of the vector if the above code is followed by the statement:

myList.at(0) = myList.at(2);

1. 10 85 10 0
2. 63 85 10 0
3. 63 85 63 0

**[Constructors]**

**Question 10: (6 points)**

**The following program will test your knowledge of when base class and derived class constructors and destructors are called.**

**Consider the following code and determine the output.**

#include <iostream>

using namespace std;

// Base class

class BaseDemo{

public:

BaseDemo() // Constructor

{ cout << "This is the BaseDemo constructor.\n"; }

~BaseDemo() // Destructor

{ cout << "This is the BaseDemo destructor.\n"; }

};

// Derived class

class DeriDemo : public BaseDemo{

public:

DeriDemo() //Constructor

{ cout << "This is the DeriDemo constructor.\n"; }

~DeriDemo() // Destructor

{ cout << "This is the DeriDemo destructor.\n"; }

};

int main(){

cout << "We will now create a DeriDemo object.\n";

DeriDemo object;

cout << "The program is now going to end.\n";

return 0;

}

OUTPUT:

**[Inheritance]**

**Question 11:(6 points)**

#include <iostream>

using namespace std;

// Base class

class Dog

{

protected:

double weight;

public:

Dog(double w)

{ weight = w; }

void bark( ) const

{

cout << "I am a dog weighing "

<< weight << " pounds." << endl;

}

};

// A SheepDog is a special type of Dog

class SheepDog:public Dog

{

int numberSheep;

public:

SheepDog(double w, int nSheep) : Dog(w)

{

numberSheep = nSheep;

}

void bark( ) const

{

cout << "I am a sheepdog weighing "

<< weight << " pounds \n and guarding "

<< numberSheep << " sheep." << endl;

}

};

int main( )

{

// Create an array of dogs

const int NUM\_DOGS =3;

Dog \*kennel[ ] = { new Dog(40.5),

new SheepDog(45.3, 50),

new Dog(24.7)

};

// Walk by each kennel and make the dog bark

for (int k = 0; k < NUM\_DOGS; k++)

{

cout << k+1 << ": ";

kennel[k]->bark( );

}

return 0;

}

Below is the output of the above code.

1: I am a dog weighing 40.5 pounds.

2: I am a dog weighing 45.3 pounds.

3: I am a dog weighing 24.7 pounds.

This is not what I was expecting.

Explain how this code can be fixed so that the output will be the following:

1: I am a dog weighing 40.5 pounds.

2: I am a sheepdog weighing 45.3 pounds

and guarding 50 sheep.

3: I am a dog weighing 24.7 pounds.

**[Abstract Classes]**

**Question 12:(4 points)**

Describe what make a class in C++ an abstract class and give an example.