**CPSC 1020 SUMMER 2016**

**EXAM #2**

**There is a total of 111 points. (11 points extra credit) Good Luck!!**

**Question 1: This covers the next 9 questions. Each question has a point value for a total of 45 points**

**Write the implementation for the Array class. You will not implement all of the class, just the functions indicated.**

#include <iostream>

using namespace std;

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;}

};

**You will NOT implement this constructor**

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

**\*The default constructor**

**\*increment static variable**

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

Array::Array()

{

We will assume this constructor sets size to 10.

Allocates the appropriate amount of memory, initializes the elements of the array to 0,

and **increments the static variable by 1**.

}

**5 points Implement this constructor**

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

**\*The regular constructor**

**\***

**\*increments the static variable**

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

Array::Array(int size)

{

}

**5 points Implement this constructor**

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

**\*The copy constructor**

**\***

**\*increments static variable**

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

Array::Array(const Array &obj)

{

}

**5 points Implement the destructor**

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

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

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

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

Array::~Array()

{

}

**3 points Implement the getter**

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

**\*The getter for size**

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

int Array::getSize() const

{

}

**5 points Implement the operator=**

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

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

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

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

{

}

**5 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

{

}

**5 points**

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

int main()

{

Array arr1(5);

arr1.setElement(0,10);

arr1.setElement(1,20);

cout << "arr1 ptr[0] is: ";

arr1.print(0);

cout << endl;

cout << "arr1 ptr[1] is: " ;

arr1.print(1);

cout << endl;

Array arr2 = arr1;

cout << "arr2 ptr[0] is: ";

arr2.print(0);

cout << endl;

cout << "arr2 ptr[1] is: ";

arr2.print(1);

cout << endl;

Array arr3;

arr3 = arr1;

cout << "arr3 ptr[0] is: ";

arr3.print(0);

cout << endl;

cout << "arr3 ptr[1] is: ";

arr3.print(1);

cout << endl;

cout << Array::getArrayCount() << endl;

return 0;

}

**12 points**

What is the output of this program:

**Question 2: (10 points) Dynamic Memory Allocation**

Given the following pointer declaration, dynamically allocate memory for this pointer.

double \*dPtr:

Suppose I want to dynamically allocated memory for an array of doubles of size 100. Write the code to dynamically allocate the memory for this array of doubles:

int size = 100;

double \*dPtr2;

When your program is done with this memory you should return the memory to the operating system. For the two questions above, write the code to return the allocated memory to the operating system.

**Question 3: (3 points)**

Fill in the blank.

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

**Question 4: (10 points) Member Initialization List**

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

class Date{

private:

string month;

int day, year;

public:

Date(string m, int d, int y){month = m; day = d; year = y}

Date(){month = " "; day = 0; year = 0;}

void setMonth(string m){month = m;}

string getMonth() { return month; }

};

**/\*Write the constructor below using member initialization list to initialize the Date and Person class member variables Note: this is NOT an inheritance problem. Person “has a” Date. \*/**

class Person

{

private:

string name;

Date dateOfBirth;

public:

**THIS IS THE CONSTRUCTOR YOU ARE TO COMPLETE**

Person(string name, string month, int day, int year)

string getMonth() { return dateOfBirth.getMonth();}

};

**Question 5: (8 points)**

What is a dangling pointer and how can you fix it:

What causes a memory leak and how can you fix it:

**Question 6: (5 points)**

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

**Question 7: (5 points) True / False**

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

**Question 8: (10 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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_

setA \_\_\_\_\_\_\_\_\_\_\_

setB \_\_\_\_\_\_\_\_\_\_\_

setC \_\_\_\_\_\_\_\_\_\_\_

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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_

setA \_\_\_\_\_\_\_\_\_\_\_

setB \_\_\_\_\_\_\_\_\_\_\_

setC \_\_\_\_\_\_\_\_\_\_\_

**Question 8: (5 points) Consider the following classes**

class Person

{

private:

string name;

public:

Person();

Person(string pName);

void setName(string pName);

string getName() const;

};

class Student:public Person

{

private:

Discipline major;

Person \*advisor;

public:

Student(string sname, Discipline d, Person \*adv);

void setMajor(Discipline d);

Discipline getMajor() const;

void setAdvisor(Person \*p);

Person \*getAdvisor() const;

};

class Faculty:public Person

{

private:

Discipline department;

public:

Faculty(string fname, Discipline d);

void setDepartment(Discipline d);

Discipline getDepartment( ) const;

};

class TFaculty: public Faculty

{

private:

string title;

public:

TFaculty(string fname, Discipline d, string title);

void setTitle(string title);

string getName( ) const;

};

Draw a chart that represents the hierarchy of these classes. Label the base classes and the derived classes.

**Question 9: (10 points)**

**Consider the following class definition:**

#include <iostream>

using namespace std;

class NumberArray

{

private:

double \*aPtr;

int arraySize;

public:

NumberArray operator=(const NumberArray &right); **operator=**

NumberArray(const NumberArray &); **copy constructor**

NumberArray(int size, double value); **regular constructor**

~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.

#include <iostream>

#include <iomanip>

//#include "NumberArray2.h"

#include "overload2.h"

using namespace std;

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

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

return 0;

}