**CS273 ASSIGNMENT #1: C++ Review**

## NAME: DUE: 26 May 2016

**GRADE:**

|  |  |  |
| --- | --- | --- |
| **CATEGORY** | **POINTS** |  |
| **EX1\_01: Pointers** |  | **25** |
| **EX1\_02: Classes** |  | **25** |
| **EX1\_03: Templates/STL Vectors** |  | **25** |
| **EX1\_04: UML/Operator Overloading** |  | **25** |
| **TOTAL** |  | **100** |

**Objectives:**

* Review important concepts learned in Computer Science II (e.g. CS172)
* Prepare you for the task of implementing the advanced concepts and abstractions that we will be studying in CS273.
* **Remind you that the best way to learn a programming concept is to implement a small test program that uses the concept.**

**Instructions:**

**This may be one of the most important homework assignments you complete this semester**. It is important to do your best to complete this well, as everything we will be learning this semester will build upon the concepts covered in this assignment!

**Homework pre-requisite:**

* + **Please read Chapter P and sections 1.3 of your textbook.** We will not be going through this in class since you have all completed CS171 and CS172
  + **Please read appendix B.1 of your textbook to review how UML diagrams are drawn.**
  + It will also be very helpful to refer back to your CS172 textbook and homework assignments.

**For most of the questions – please** **write a test program to verify that what you are writing actually works**. I will deduct points from obvious syntax errors. Please allow Visual Studio to help you with this homework!

**EX1\_01: Pointers**

1. Write the **C++ declaration for a pointer variable** that can store the **address of a double** variable:

double \*pointVariable;

1. Write a C++ statement that will use the C++ **new** operator to **dynamically** **allocate memory** for a double variable. Store the address of this newly allocated double in the **pointer variable** you declared in the previous step.

double\* newPointVariable = new double;

pointVariable = newPointVariable;

1. Write a C++ statement that will **assign the double value 4.12** to the memory for the double variable that you allocated from step 5. **HINT**: You will need to use the **\*** operator and the pointer variable that holds the address of the double.

\*newPointVariable = 4.12;

1. Write a C++ statement that will **release the memory** that was allocated using the **new** operator.

delete newPointVariable;

1. What is wrong with the following code?

char \*variable ;

variable = 3;

doesn’t store 3 because you would need to use “ \* “ in order for it to be stored.

1. What operator would you use to obtain the **memory address** of the variable, **myVar**?

**int myVar;**

for (int i = 0; i < 10; i++)

{

elements[i] = 42;

elements++;

}

return 0;

1. **Declare and allocate** an array of 10 **int** elements using **dynamic memory allocation**.

delete myVar;

1. **Write a C++ for loop** that assigns the integer value 42 to every element in the array allocated in the previous question. **However, you must use** **pointer arithmetic to advance and access each element** (don’t use array notation).

General :

For(int i=0; I < elements.size(); i++;) // the object has a function called size that returns the size of the array..

{

elements[i] = 42;  
}

Pointer Arithmatic:

for (int i=0;  I < 10; i++) begins with 0 and ends with 10

{

 elements\*[i]=42

}

1. **Write a statement that will release the memory** allocated for the array in question 6 back to the operating system.

delete myVar;

1. **Write the code for a function** that has the following prototype

int \*doubleCapacity(int \*list, int size);

The function doubles the **size** (indicated by the parameter ***size***) of the array passed in through parameter ***list***.

(HINT: This was a CS172 homework assignment!)

// the array of the double capacity is passed to it and copies the elements

int doubleCapacity(const int\* list, int size)

{

//double size

int \*list2 = new int[2 \* size];

for (int i = 0; i < size; i++)

{

list2[i] = list[i];

}

delete[] list; // removes the old

return \*list2;

}

**EX1\_02: Classes**

1. **Declare a C++ class** called **Circle** that conforms to the following description:
   1. It has a **private** **property (member variable)** called **radius** that is of a **double** data type. ()
   2. It has a **public** **default constructor** that initializes the **radius** of a circle object to “0” ()
   3. It has a **public** **overloaded constructor** that takes a parameter to initialize the **radius** of a circle object ()
   4. It has a **public method called getArea()** that returns the computed area (π\*radius\*radius) of the circle object. ()
   5. It has **public** **getter** and **setter** methods for accessing and modifying the radius of a circle object respectively ()
2. **Declare** a **Circle** object variable, called **myCircle1,** using the **default constructor**.
3. **Declare** a **Circle** object variable, called **myCircle2,** using the **overloaded constructor** to initialize its radius to “10”.
4. **Declare a pointer variable** to a **Circle** object, and allocate memory for it using the overloaded constructor to initialize its radius to “12”.
5. **Declare an array** of 10 **Circle** objects using the default constructor.
6. **Write a C++ for loop** that assigns the radius of every Circle object in the array defined in the previous question to “15”.

**EX1\_03: Templates/STL Vector:**

1. **Using templates, convert** the following Swap function to work with arguments of any generic type T:

void Swap(int & A, int & B) {

int tmp = A;

A = B;

B = tmp;

}

1. **Using templates, convert** the following class to hold an array of any generic type T:

class MyVec {

private:

int \*array; // dynamically allocated array

public:

MyVec(int size) { // constructor creates array of size “size”

array = new int[size];

}

~MyVec() { // destructor returns memory back to system

delete [] array;

}

};

1. **Declare an object of the template class** defined in the previous question (i.e. **MyVec**) to hold an array of **double** data types.

MyVec<double> (36);

1. **Declare an STL vector** object for storing elements of the **char** data type.

vector <char> cVector[10];

1. **Write a C++ for** loop to assign a value (of your choice) to every element in the vector object declared in the previous question.

for (int i=0; I <10; i++)

{

cVector[i].push\_back(‘t’);

1. **What STL vector method** do you use to get the current size of the vector?

**cVector.size**

## EX1\_04: UML/Operator Overloading

1. **Draw a UML diagram** describing the composition relationship between the Elevator and Building class.

**class Elevator {**

**…**

**};**

Elevator

Building

**class Building {**

**public:**

**Elevator elevator;**

**…**

**};**

1. **Draw a UML diagram** describing the inheritance relationship between the Square and Shape class.

**class Shape {**

Square

**…**

Shape

**};**

**class Square : public Shape {**

**…**

**};**

1. When class A **inherits** class B, class A also inherits all public methods and variables in class B? (true/false)

**true**

1. When class A **inherits** class B, can the **private** members variables of class B be accessed in class A? (yes/no)

**YES**

1. **Write C++ code for the classes “Animal”, “Cat”, and “Mouth”.** These classes must implement the following UML relationship, i.e. “Cat” is an “Animal”, and it has a “Mouth”.

The C++ classes do not need to have any other member variables or methods, other than what is needed to demonstrate the class relationship.

**Create a visual studio C++ project and define you classes in there.**

Your main() function should declare a **Cat** object.

**Email the source code files (.cpp and .h) to your instructor**

Animal

Cat

Mouth

1. Create a Visual Studio C++ project and **define the Circle class** used in EX1\_02. For the class, **define an operator + method** that will add 2 circles together to give you a new Circle with their radius added.

Your main() function should declare 2 circle objects of different radii. Add the 2 circles with “+” to derive a 3rd circle object with a combined radii.

**Email the source code files (.cpp and .h) to your instructor**

**(Operator overloading is a concept we will be using throughout this semester. Please review page 31 of your CS273 textbook, and look back to your CS172 book as well, or look online.)**