**CPSC 1020 Summer 2016**

**FINAL EXAM**

**This is a Closed Book exam. Please keep your notes and your computers closed. You have 2.5 hours to complete the Exam. There is a total of 112 points with 12 extra credit**

**[Definitions]**

**Question 1: (1 point each, for a total of 20 points) There are more definitions than words.**

**Matching**

1. Base Class 12. Binary Search
2. Derived Class 13. Recursion Function
3. Dynamically Allocated Memory 14. Polymorphic
4. Pass by Value 15. Aggregation and Composition
5. Pass by Reference 16. Operator Overloading
6. Inheritance 17. Self-referential Pointer
7. Override of Function 18. Memory Leak
8. Virtual Function 19. Dangling Pointer
9. Size 20. showpoint
10. STL
11. Linear Search

\_\_\_\_\_ A function parameter that points to another variable and any change made to the parameter in the function is made to the variable it points to.

\_\_\_\_\_ describes a general class that other classes can inherit data members and functions, also known as a parent class

\_\_\_\_\_ An algorithm that finds the position of a target value within a sorted array.

\_\_\_\_\_ When an argument is passed into a parameter, only a copy of the argument’s value is passed. Changes to the parameter do not affect the original argument.

\_\_\_\_\_ A data structure that may be implemented as a linked list and expand or shrink with each addition or deletion operation.

\_\_\_\_\_ Causes a decimal point and trailing zeros to be displayed for floating-point numbers, even if there is no fractional part

\_\_\_\_\_ Used by C++ to redefine how standard operators work when used with class objects.

\_\_\_\_\_ A data structure that has a fixed size and is implemented using an array.

\_\_\_\_\_ A data structure that stores and retrieves items in a first-in-first-out manner

\_\_\_\_\_ Code that produces different behavior when executing code with different types.

\_\_\_\_\_ A function that calls itself.

\_\_\_\_\_ Returns the number of elements in a vector.

\_\_\_\_\_ This is when a derived class provides functionality of a function with the same name, same parameters, and same return value of a function in the base class.

\_\_\_\_\_ describes a class that inherits functionality and data members from another class

\_\_\_\_\_ Allows us to define a class in terms of another class. Creates an “is-a” relationship.

\_\_\_\_\_ A library provided by C++ that is a collection of programmer-defined data types and algorithms that are available for you to use in your C++ programs.

\_\_\_\_\_ Contains the address of memory that has been given back to the operating system

\_\_\_\_\_ A data structure that stores and retrieves items in a last-in-first-out manner.

\_\_\_\_\_ Allows the most specific version of a member function in an inheritance hierarchy to be selected for execution. This is what makes polymorphism possible.

\_\_\_\_\_ this is when a program, while running, asks the computer to set aside a chunk of unused memory large enough to hold a variable of a specific data type.

\_\_\_\_\_ A data structure pointer that points to an object of the same data structure type

\_\_\_\_\_ Occurs when an object of one class owns an object of another class (has-a) relationship.

\_\_\_\_\_ A search algorithm that uses a loop to sequentially step through an array, starting with the first element.

\_\_\_\_ Sets a minimum print field width of size n for the next value output

\_\_\_\_\_ This is what occurs when you are no longer using dynamically allocated memory and do not give the memory back to the operating system.

**Question 2: True/False: (1 point each for a total of 7)**

\_\_\_\_\_ Objects can be passed to functions, but they cannot be returned by functions.

\_\_\_\_\_ When an object is passed to a function, but the function is not supposed to change it, it is best to pass the object as a reference.

\_\_\_\_\_ An abstract class cannot be instantiated.

\_\_\_\_\_ The following is an example of how a pure virtual function is declared?

class MyClass

{

   public:

      virtual myFunction( ) = 0;

{

cout << "Hello"

};

};

\_\_\_\_\_ A reference can be initialized at any time, but a pointer must be initialized when created.

\_\_\_\_\_ Considering an linked list: If the self-referential pointer in the last node has a value of NULL this signifies the end of a linked list?

\_\_\_\_\_ One advantage a vector has over a linked list is it is easier to insert an element anywhere in a vector than it is to insert a node anywhere in a linked list.

**[Manipulators]**

**Question 3: (4 points):**

Given the following program, what is the output.

OUTPUT:

#include <iostream>       
#include <iomanip>        
using namespace std;  
  
int main () {  
  double f =3.14159;  
  cout << setprecision(5) << f << endl;  
  cout << setprecision(9) << f << endl;  
  cout << fixed;  
  cout << setprecision(5) << f << endl;  
  cout << setprecision(9) << f << endl;  
  return 0;  
}

[**Type Compatibility]**

**Question 4: (2 points)**

Suppose that the classes **Dog** and **Cat** derive from **Animal**, which in turn derives from **Creature.** Suppose further that **pDog, pCat, pAnimal,** and **pCreature** are pointers to the respective classes.  Suppose **Animal** and **Creature** are both abstract classes.

The following code is incorrect. Rewrite the code so that if incorporated in a program it would not cause a compile error.

pAnimal = new Dog;

pDog = pAnimal;

**[STL List]**

**Question 5: (1 point)**

Using the STL list class, declare a list of pointers of type **Animal** called **zoo**.

**[STL vector]**

**Question 6 - 9: (2 point each for a total of 8 points)**

Define a variable called **tigers** to be an empty vector of **ints**.

Define a variable called **bears** to be a 10-element vector of doubles.

Write a statement that stores the value 27 in **tigers.**

Write a statement that stores the value of 12.789 in element 4 of **bears**.

**[Copy Constructor]**

**Question 10: (2 points)**

Which of the following describe when a copy constructor is called: There may or may not be more than one choice.

A.  When an object is initialized with another object's data

B.  When an object is passed by reference to a function

C. When an object is returned by value

D. None of the above

**[C-style pointer]**

**Question 11: (2 points)**

What is the error in the following snippet of code?

int \*x;

int y;

x = y;

**[C++ Reference]**

**Question 12: (2 points)**

Assume the following:

int var = 5;

Which of the following is a legitimate declaration of a C++ reference variable?

1. &int refvar;
2. &int refvar = 5;
3. int& refvar;
4. int& refvar = var;
5. none of these

**[C-style Pointer]**

**Question 14: (6 Pointers)**

**What is the output of the following program?**

#include <stdio.h>

int main()

{

OUTPUT:

int values[10] = {-1, 14, -24, 6, 9, 2, -3, 4, 7, 3};

char word[26] = {'H', 'a', 'v', 'e', ' ', 'a', ' ','g', 'r', 'e',

'a', 't', ' ', 'w', 'e', 'e', 'k', 'e', 'n', 'd', '!', '!', '!', '!'};

int i, n=10;

int \*pv0 = values;

printf("v0 = %i\n", \*pv0);

int \*pv1 = pv0 + 3;

printf("pv1 = %i\n", \*pv1);

char \*pv2;

pv2 = word;

printf("\*pv2 = %c\n", \*pv2);

printf("\*(pv2 + 3) = %c\n\n", \*(pv2 + 3));

// output

char \*pv4 = word;

for (i=0; i<5; i++) {

printf("%c", \*pv4);

pv4 += 3;

}

printf("%s\n", word);

return(0);

}

**[C++ Classes]**

**Question 15: This is a multi-part question. First, fill in the code as instructed in each function below. Second, according to the code in main give the output of the code in the box provided.**

**This class creates an array of doubles and sets each element in the array to a value passed into the constructor**

#include <iostream>

using namespace std;

class NumberArray

{

private:

double \*aPtr;

int arraySize;

public:

NumberArray operator=(const NumberArray &right); //overloaded operator function

NumberArray(const NumberArray &);

NumberArray(int size, double value);

~NumberArray();

void print();

void setValue(double value);

};

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

//Destructor

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

NumberArray::~NumberArray()

{ cout << "Destructing" << endl;

**//WRITE THE CODE FOR THE DESTRUCTOR 3 points**

}

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

//The overloaded operator function for assignment

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

NumberArray NumberArray::operator=(const NumberArray &right)

{ cout<< "In operator =" << endl;

cout<< "In operator =" << endl;

//YOU CAN ASSUME THIS CODE IS ALREADY WRITTEN

cout << "Return in =" << endl;

return \*this;

}

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

//Copy Constructor

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

NumberArray::NumberArray(const NumberArray &obj)

{ cout << "In copy constructor" << endl;

**//WRITE THE CODE FOR THE COPY CONSTRUCTOR 6 points**

}

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

//Constructor

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

NumberArray::NumberArray(int size, double value)

{ cout << "In regular constructor" << endl;

**//WRITE THE CODE FOR THE REGULAR CONSTRUCTOR 6 points**

}

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

//Sets the value stored in all entries of the array

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

void NumberArray::setValue(double value)

{

for(int index = 0; index < arraySize; index++)

aPtr[index] = value;

}

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

//Print out all entries in the array

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

void NumberArray::print()

{

for(int index = 0; index < arraySize; index++)

cout << aPtr[index] << " ";

OUTPUT:

cout << endl;

}

**What is the output of the main? 15 points**

int main()

{

NumberArray first(3, 10.5);

first.print();

NumberArray second(5, 20.5);

second.print();

NumberArray third(7, 30.5);

third.print();

first = second = third;

return 0;

}

**[Bitwise Operation]**

**Question 16: (9 points)**

Consider the following:

**Show your work.** You will need to convert a, b, and c to binary, do the bitwise operation then convert the answer back to decimal.

int a = 45;

int b = 93;

int c = 15;

int d = a|b&c;

What is the value of d?

1. 47
2. 13
3. 15
4. 45

**[Exception Handling]**

**Question 17: (2 point)**

There is an error in this block of code.  Describe the error.

catch{

   quotient = divide(num1, num2);

   cout << "The quotient is " << quotient << endl;

}

try(string exceptionString){

   cout << exceptionString;

}

**[Templates]**

**Question 18: (2 points)**

There is an error in the following block of code.  Describe the error.

template <class T>

T square (T number)

{

return (T \* T);

}

**[Member Initialization and Operator Overloading]**

**Question 19: 15 Points**

**Consider the following class. Implement the functions indicated below.**

#ifndef POINT\_H

#define POINT\_H value

#include "ostream"

#include "iostream"

class Point {

private:

int x,y;

public:

Point ();

Point (int x, int y);

Point (const Point& p);

virtual ~Point ();

int get\_x() const;

int get\_y() const;

void set\_x(int);

void set\_y(int);

// + operator

Point operator+(const Point& rhs) const;

// - operator

Point operator-(const Point& rhs) const;

// = operator

Point& operator=(const Point& rhs);

// ostream

friend std::ostream& operator<<(std::ostream& out, const Point& p);

};

#endif

#include "Point.h"

using namespace std;

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

\*Complete the constructors using Member Initialization List Technique

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

Point::Point ()

Point::Point (int x, int y)

Point::Point (const Point& p)

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

\*You do not need to implement the destructors nor the getters

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

Point::~Point () {}

int Point::get\_x() const { return x; }

int Point::get\_y() const { return y; }

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

\*Implement the setters

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

void Point::set\_x(int x)

void Point::set\_y(int y)

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

\*Implement the +, -, and = operators

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

// + operator

Point Point::operator+(const Point& rhs) const

{

}

// - operator

Point Point::operator-(const Point& rhs) const

{

}

// = operator

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

{

}

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

\*Implement the << operator such that it will print something

\*similar to the following

\*[3,4] when the operator << is called using a Point object

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

// ostream

std::ostream& operator<<(std::ostream& out, const Point& p)

{

}