**CPSC 1020 Spring 2018**

**Final Exam**

**This is a Closed Book exam. Please keep your notes and your computers closed. You have 2 hours and 10 minutes to complete the Exam. There is a total of 59 points. Although the points total 59 your grade will reflect 100%**

**Question 1: [Matching] (26 Points)**

1. **PolyMorphic 14. Pass by Reference**
2. **STL 15. Head**
3. **Left or Right 16. Size**
4. **Setprecision(n) 17. Encapsulation/data hiding**
5. **Base Class 18. Member Initialization List**
6. **Inheritance 19. Local Function**
7. **Fixed 20. Base Class Access Specifier**
8. **Composition 21. Rule of Three**
9. **Showpoint 22. Operator Overloading**
10. **Static Member Variable 23. Setw(n)**
11. **Virtual Function 24. Derived class**
12. **Dynamic Allocated memory 25. Global Function**
13. **Pass By Value 26. Protected**

\_\_\_\_\_\_ 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.

\_\_\_\_\_\_Allows us to set the number of significant digits.

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

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

\_\_\_\_\_\_ Initialized outside of the class declaration and is shared by all instances of the class.

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

\_\_\_\_\_\_ Sets the size of a print field.

\_\_\_\_\_\_ Causes a decimal point and trailing zeros to be displayed for floating-point numbers, even if there is not fractionl part.

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

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

\_\_\_\_\_\_ A variable that is defined inside a function and is not accessable outside the function.

\_\_\_\_\_\_ Points to the first node in a linked list.

\_\_\_\_\_\_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.

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

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

\_\_\_\_\_\_ Forces a floating point number to display in a particular point format.

\_\_\_\_\_\_ Something that says if you need a to write a copy constructor you need an assignment operator and destructor.

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

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

\_\_\_\_\_\_ This type of members of a base class are like private members, except they may be accessed by derived classes.

\_\_\_\_\_\_ Determines how private, protected, and public base class members are accessed when they are inherited by the derive class.

\_\_\_\_\_\_ A variable defined outside all functions and is accessible to all functions in its scope.

\_\_\_\_\_\_ Allows constructors of classes to pass arguments to constructors of member objects using a list of comma-separated calls to member object constructors. It also is used to initialize member variables of any type.

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

\_\_\_\_\_\_ One of the fundamentals of OOP. It refers to the bundling of data with the methods that operate on the date. It also is used to hide the values or state of a structured data object inside a class, preventing unauthorized parties' direct access to them.

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

**Question 2. [Static Variable] (1 Point)**

**There may be more than one correct answer.**

A static variable that is part of a class should be initialized where?

Circle all that apply:

1. In the setter
2. On the same line as the declaration
3. In the constructors
4. Outside of the class declaration
5. None of these

**Question 3. [Copy Constructor] (1 Point)**

**There could be more than one answer for this question.**

Which of the following describes when a copy constructor is called?

Circle all that apply:

1. When an object is passed by reference to a function.
2. When an object is initialized with another objects' data.
3. When an object is returned by value.
4. When an object is returned from a function by reference.
5. All of these
6. None of these

**Question 4 [Double Pointers] (1 Point)**

Consider the following snippet of code:

int first = 10;

int second = 15;

int third = 20;

int \*sptr = &first;

int \*sptr2 = &second;

int \*\*dptr = &sptr;

\*dptr = sptr2;

After the following statement:

             \*dptr = &third;

What will the following print statement print?

printf("%d", \*\*dptr);

Circle one:

1. 20
2. 10
3. 15
4. None of these because you cannot print this way.

**Question 5 Dynamic Memory (1 Point)**

Assume we have dynamically allocated the memory for a pointer, called **pix**, to 10 objects of type **Pixel**.  I am now done with this memory and need to return the memory to the operating system.  In C++ which of the following is the appropriate way to give the memory back to the operating system?

Circle one:

1. delete Pixel [];
2. free(pix)
3. delete [] pix
4. pix.delete();
5. ~Pixel()

**Question 6 Vector (1 Point)**

Assume I created an empty vector of type integer and called it tiger.  Now I want to add an element with the value of 15 to the vector.  Which of the following is the appropriate way to add the element to the vector?

1. tiger[0].push\_back(15);
2. tiger.push\_back(15);
3. tiger.at(0).set(15);
4. tiger(15);

**Question 7 C-Pointers (1Point)**

Consider the following program. What is the output of this program?

int main()

{

            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', ' ', 'S', 'u', 'm', 'm', 'e', 'r', '!', '!', '!', '!', '!'};

            int i, n=10;

             int \*pv0 = values;

            printf("%i", \*(pv0+3));

            char \*pv1 = word;

            for (i=0; i < 4; i++)

            {

                        printf("%c", \*pv1);

                         pv1 += 5;

            }

             return(0);

}

Circle the correct output:

1. -24H am
2. 6Haam
3. -24Haam
4. 6H am
5. None of these

**Question 8 Copy Constructor (5 points)**

Consider the NumberArray class on Attachment 1.

You are going to fill in the blanks necessary to write the code for a copy constructor for the NumberArray class.  There may be more blanks than needed. Choose the appropriate choice for each blank. **If there should not be anything in a blank then choose none**.

NumberArray::NumberArray(NmberArray &obj)

{

       \_\_\_\_\_\_\_

       \_\_\_\_\_\_\_

       \_\_\_\_\_\_\_

       \_\_\_\_\_\_\_

       \_\_\_\_\_\_\_

}

Choose one for each blank:

1. aPtr = new double[arraySize];
2. aPtr = new NumberArray[];
3. arraySize = obj.arrySize;
4. arraySize = getSize();
5. aPtr = new double;
6. for(int index = 0; index < arraySize; index++)
7. \*aPtr = \*obj.aPtr;
8. \*aPtr = obj->aPtr;
9. aPtr[index] = obj.aPtr[index];
10. return \*this;
11. return this;
12. None

**Question 9 Abstract Classes (1 Point)**

Circle true or false

True / False

Assume you have an abstract class called MyBaseClass and a derived class of MyBaseClass called MyDerivedClass.  The following is an example of how to create an instance of MyBaseClass?

MyBaseClass \*ptr = new MyDerivedClass;

**Question 10 Abstract Classes (1 Point)**

Circle true or False

True / False

The following is an example of an abstract class.

class MyClass

{

    public;

      virtural myFunction() { cout << "Hello" << endl; }

}

**Question 11 Pointers (1 Point)**

Circle true or false

True / False

The following is valid code?

int \*num;

cout << \*num << endl;

**Question 12 [Pointers] (1 Point)**

Circle true or false

True/False

The following is valid code?

float \*fPtr;

int num;

fPtr = &num;

**Question 13 [What is the output] (6 points)**

Using the Date class, shown in Attachment 2, what is the output.  Fill in the blank with one of the choices to the right.  There may be more lines for output than needed.  If so choose none.

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

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

Default

Regular

Copy

Operator=

Destructor

doSomething

none

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

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

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

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

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

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

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

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

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

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

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

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

**Question 14 [OutPut] (1 Point)**

Circle the correct answer from the list below:

What is the output of the following program?

#include <stdio.h>  
void f(int\*, int);

int main()  
{  
int i = 8, j = 10;  
f(&i, j);  
printf("%d\n", i+j);  
return 0;  
}

void f(int\* p, int m)  
{  
m = m + 5;  
\*p = \*p + m;  
}

Chose one:

1. 17
2. 30
3. 33
4. 36

**Question 15 [Constant variable and #define**

Circle true or false

True / False

In class we discussed one major difference between C++ constant variables and C-style #define. The difference is a constant variable does not have a memory location but a #define does.

**Question 16[Function Pointer] (3 Points)**

Consider the following code and circle from the list the appropriate answer for each of the three lines of code.

#include <stdio.h>

void printNum(int);

void numSq(int);

int adder(int, int);

int main(int argc, char\* argv[]){

//1. Create a function pointer called fp that will point to a function that passes in 2 ints as parameters.

1. \*fp(int, int);
2. int(\*fp)(int, int);
3. int \*fp(int, int);
4. int(int, int)\*fp;
5. None of these

              //Point the function pointer created above to the adder function.

1. adder = fp;
2. &fp=adder;
3. fp = adder;
4. \*adder = fp;

              //Call the function pointer with the values of 5 and 10, returning the sum to the variable sum.

1. int sum = fp(5,10);
2. double sum = \*fp(5,10);
3. int sum = (\*fp)(5,10);
4. int sum = (fp, 5, 20)

              printf("Sum = %d\n", sum);

              return 0;

}

void printNum(int n){ printf("The number is %d\n", n);}

void numSq(int n){printf("The square of n (%d) is %d\n",n, n\*n);}

int adder(int a, int b){ return (a + b);}

**Question 17 Inheritance (1 Point)**

With respect to inheritance:

When both a base class and a derived class have constructors, the base class's constructor is called \_\_\_\_\_\_\_\_.

Circle the correct answer:

1. first
2. last
3. never

**Question 18 Inheritance (1 Point)**

With respect to inheritance:

When both a base class and a derived class have destructors, the base class's destructor is called \_\_\_\_\_\_\_\_\_\_\_.

Circle the correct answer:

1. first
2. last
3. never

**Question 19 Reference variable (1 Point)**

Assume the following:

int var = 5;

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

Circle one:

1. &int refvar;
2. &int refVar = 5;
3. int & refvar;
4. int& refvar = var;

**Question 20 Output (4 Points)**

For each of the following output lines choose the correct output:

Given the following program what is the output?

#include <iostream>

#include <iomanip>

using namespace std;

int main () {  
  double f =3.14159;  
  cout << setprecision(5) << f << endl;

Circle one:

1. 3.14159
2. 3.1416
3. 3.14160
4. 0.314159
5. 0.31416

  cout << setprecision(9) << f << endl;

Circle one:

1. 3.14159
2. 3.14159000
3. 3.141590000
4. 0.314159000

  cout << fixed;  
  cout << setprecision(5) << f << endl;

Circle one:

1. 3.14160
2. 3.1416
3. .0314159
4. 3.14159

  cout << setprecision(9) << f << endl;

Circle one:

1. 3.141590000
2. 3.14159000

  return 0;  
}