**CPSC 1020 SUMMER 2016**

**FINAL EXAM**

**Question 1: ( 12 points )**

Consider the following class. In main write the code that will print the value of gallons for the three variables declared. Write the programs output in the box to the right.

#include <iostream>

using namespace std;

class Tank

{

Output: (9 points)

private:

int gallons;

public:

Tank()

{ cout << “Default constructor” << endl;

gallons = 50;

}

Tank(int gal)

{ cout << “Regular constructor” << endl;

gallons = gal;

}

int getGallons()

{ cout << “Returning gallons” << endl;

return gallons;

}

};

int main( )

{

Tank storage1, storage2, storage3(20);

**//YOUR CODE GOES HERE**

**//(3 point)**

return 0;

**}**

**Question 2: (5 points )**

Write a class declaration for a class named Circle, which has a private data member radius, of type double, and public member function setRadius and getArea. Implement the constructors and member functions as inline functions. Hint: area of a circle is r2.

**Question 3: (30 points )**

Consider the following class definition:

#include <iostream>

using namespace std;

class IntArray

{

private:

int \*aptr;

int arraySize;

void subError() const; // Handles subscripts out of range

public:

IntArray(int); // Constructor

IntArray(const IntArray &); // Copy constructor

~IntArray(); // Destructor

int size() const{ return arraySize; }

int &operator[](int) const; // Overloaded [] operator

};

You are to implement this class:

#include "intarray.h"

#include <cstdlib>

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

// Constructor for IntArray class. Sets the size of

// the array and allocates memory for it. Initialize the array elements

// to 0.

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

(8 points)

IntArray::IntArray(int s)

{

}

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

// Copy constructor for IntArray class.

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

//(8 points)

IntArray::IntArray(const IntArray &obj)

{

}

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

// Destructor for IntArray class.

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

//(6 points)

IntArray::~IntArray()

{

}

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

// subError function. Displays an error message and

// exits the program when a subscript is out of range.

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

void IntArray::subError() const

{

cout << "ERROR: Subscript out of range.\n";

exit(0);

}

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

// Overloaded [] operator. The argument is a subscript. If the subscript

// is out of bounds it calls the subError function otherewise,

// this function returns a reference to the element

// in the array indexed by the subscript.

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

//(8 points)

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

{

}

**Question 4: (8 points)**

Consider the provided Date class definition, implementation, and provided main. When compiled and ran what will this program output.

Output:

**Question 5: (2 points)**

BitWise Operations  Masking

Masking is commonly used to determine the value of one or more individual bits within a set of other bits.

Suppose I want to determine the last three bits of the following byte.

00010101

Which would accomplish this task?

1. 0b00010101 ^ 3
2. 0b00010101 & 7
3. 0b00010101 & 2
4. 0b00010101 | 7

**Question 6: (2 points)**

Consider the following:

int main()

{

   unsigned int a = 0b00111100;

   unsigned int b = 0b00001101;

   int c = 0;

   c = a | b;

   printf("The value of c is %d", c);

   return 0;

}

What will print?

**Question 7: (2 points)**

Consider the following:

int main()

{

   unsigned int a = 0b00111100;

   unsigned int b = 0b00001101;

   int c = 0;

   c = a ^ b;

   printf("The value of c is %d", c);

   return 0;

}

What will print?

**Question 8: (2 points)**

Consider the following:

int main()

{

   unsigned int a = 0b00111100;

   int c = 0;

   c = a<< 2;

   printf("The value of c is %d", c);

   return 0;

}

What will print?

**Question 9: (2 points)**

Consider the following snippet of code:

int val;

double num = 3.7;

C-style casting looks like the following:

val = (int) num;

Re-write the above code using the post-standard C++ style of type casting.

**Question 10: (3 points)**

Consider the following program:

using namespace std;  
  
int mystery(const char \*s)  
{  
    int x;  
  
    for(x = 0; \*s != '\0'; s++)  
    {  
        x++;  
    }  
  
    return x;  
}  
  
int main()  
{  
    char string[6] = “Hello”   
  
    cout << mystery(string) << endl;  
    return 0;  
}

What is the output of this program?

**Question 11: (2 points)**

What is the error in the following snippet of code?

int \*number;

cout << \*number<< endl;

**Question 12: (2 points)**

What is the error in the following snippet of code.

float \*floatPtr;

int \*intPtr;

intPtr = floatPtr;

**Question 13: (2 points)**

What is the error in the following snippet of code?

int \*x;

int y;

x = y;

**Question 14: (2 points)**

Consider the following code:

int addInt(int a, int b)

{

      return (a + b);

}

Write the code that declares a function pointer called **ptrf** and point it to the addInt function provided.

**Question 15: (2 points)**

#include <iostream>  
using namespace std;  
  
namespace foo  
{  
  int value() { return 5; }  
}  
namespace bar  
{  
  const double pi = 3.1416;  
  double value() { return 2\*pi; }  
}  
  
int main () {  
 **WRITE THE CODE NECESSARY TO PRINT THE VALUE OF** **pi.**

  return 0;  
}

**Question 16: (4 points)**

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;  
  cout << setprecision(9) << f << endl;  
  cout << fixed;  
  cout << setprecision(5) << f << endl;  
  cout << setprecision(9) << f << endl;  
  return 0;  
}

**Question 17 (2 points)**

Assuming that **soap** is an instance of the **Inventory** class, which of the following is a valid call to the **setOnHand** member function.

1. setOnHand(20);
2. soap::setOnHand(20);
3. soap.setOnHand(20);
4. Inventory.setOnHand(20);

**Question 19: (1 point)**

True or False:

Just as a class can have multiple constructors, it can also have multiple destructors.

**Question 20: (1 point)**

When an object is passed to a function, a copy of it is made if the object is

1. Passed by value
2. Passed by reference
3. Passed by constant reference
4. Any of the above

**Question 21: (1 point)**

C++ provides us with a mechanism to create a generic function that can work with different data types.

This mechanism is called a \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

**Question 22: (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;

}

**Question 23: (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);

}

**Question 24: (2 points)**

True or False

An abstract class cannot be instantiated.

**Question 25: (2 points)**

A class with a least one pure virtual member function is called a(n) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ class.

**Question 26: (4 points)**

Type Compatablility:

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.

Rewrite the following two statements to get them to compile correctly.

pAnimal = new Dog;

pDog = pAnimal;

**Question 27: (2 points)**

Using the STL: declare a list of pointers of type Animal called zoo.

**Question 28: (4 points)**

Define **tigers** to be an empty vector of ints and **bears** to be a 10-element vector of doubles.   After the vectors have been declared, write a statement that stores the value 27 in **tigers** and a statement that stores the value of 12.789 in element 4 of **bears.**

**Question 29: (2 points)**

Describe the error in the following snippet of code.

class MyClass

{

   public:

      virtual myFunction( ) = 0; { cout << "Hello"};

};

**Question 30: (2 points)**

Which of the following describe when a copy constructor is called:

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

#include <iostream>

using namespace std;

class Date

{

private:

int month;

int day;

int year;

public:

/\*Default construtor\*/

Date();

/\*Overloaded constructor\*/

Date(int, int, int);

/\*Destructor\*/

~Date();

/\*Setters\*/

void setMonth(int);

void setDay(int);

void setYear(int);

/\*Getters\*/

int getMonth();

int getDay();

int getYear();

};

/\*Implementing the Date constructors\*/

Date::Date()

{

month = 1;

day = 1;

year = 1900;

cout << "executing the default constructor" << endl;

}

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

{

month = m;

day = d;

year = y;

cout << "executing the regular constructor" << endl;

}

/\*Implementing the Date destructor\*/

Date::~Date()

{

cout << "executing the destructor" << endl;

}

/\*Implementing Setters\*/

void Date::setMonth(int m)

{

cout << "setting month" << endl;

month = m;

}

void Date::setDay(int d)

{

cout << "setting day" << endl;

day = d;

}

void Date::setYear(int y)

{

cout << "setting year" << endl;

year = y;

}

/\*Implementing getters\*/

int Date::getMonth()

{

cout << "returning month" << endl;

return month;

}

int Date::getDay()

{

cout << "returning day" << endl;

return day;

}

int Date::getYear()

{

cout << "returning year" << endl;

return year;

}

int main()

{

Date day1;

Date day2(10,7,1963);

cout<< day2.getMonth() << " is the month" <<endl;

cout<< day1.getMonth() << " is the month for day1" <<endl;

return 0;

}