**CPSC 1020 SPRING 2016**

**FINAL EXAM**

**This is a Closed Book exam. Please keep your notes and your computers closed. There is a total of 106 points (6 extra credit)**

**Question 1: (1 points)**

In C++ constructors have the same name as the

1. Class
2. Class instance
3. Program
4. None of the above

**Question 2: (1 points )**

In C++ a constructor that requires no arguments is called \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

**Question 3: (1 points )**

True or False:

Like any C++ function, a constructor may be overloaded, providing each constructor has a unique parameter list.

**Question 4: (1 points )**

True or False:

A class can have multiple destructors.

**Question 5: ( 5 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: (4 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**

**//(1 point)**

return 0;

**}**

**Question 6: (2 points )**

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 7: (2 points )**

True or False:

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

**Question 8: (2 points )**

True or False:

When an object is passed to a function, but the function is not supposed to change it, it is best to pass it by value.

**Question 9: (8 points )**

Write a class declaration for a class named Circle, which has a private data member radius, a 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 10: (7 points ):**

Write definition statements for the following three vector objects:

(2 points) frogs (an empty vector of ints)

(2 points) lizards (a vector of 20 doubles)

(3 points) toads (a vector of 100 chars, with each element initialized to ‘z’).

**Question 11: (8 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 declare, write a statement that stores the value 27 in ***tigers*** and a statement that stores the value of 12.897 in element 4 of ***bears***.

**Question 12: (5 points )**

For each of the following variable definitions, determine whether the statement is valid or invalid.

1. int ivar; \_\_\_\_\_\_\_\_\_\_\_\_

int \*iptr = &ivar;

1. int ivar, \*iptr = &ivar; \_\_\_\_\_\_\_\_\_\_\_\_
2. float fvar; \_\_\_\_\_\_\_\_\_\_\_\_

int \*iptr = &fvar;

1. int nums[50], \*iptr = nums; \_\_\_\_\_\_\_\_\_\_\_\_
2. int \*iptr = &ivar; \_\_\_\_\_\_\_\_\_\_\_\_

int ivar;

**Question 13: (6 points )**

Given the following pointer variable:

**int \*ip;**

(3 points)Write a statement that will dynamically allocate an array of 500 integers and store its address in **ip**.

(3 points)Write a statement that will free the memory you allocated in the statement above.

**Question 14: (2 points )**

Circle **all** of the following that describes when a copy constructor is called:

1. When an object is initialized with another object’s data
2. When an object is passed by reference to a function
3. When an object is returned by value
4. None of the above

**Question 15: (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.

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

(5 points)

IntArray::IntArray(int s)

{

}

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

// Copy constructor for IntArray class.

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

//(10 points)

IntArray::IntArray(const IntArray &obj)

{

}

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

// Destructor for IntArray class.

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

//(5 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.

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

//(10 points)

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

{

}

**Question 16: (4 points )**

There are two categories of recursion direct and indirect. What is the difference?

**Question 17: (5 points )**

The following recursive program has an error:

#include <iostream>

using namespace std;

void message();

int i = 0;

int main()

{

message();

return 0;

}

void message()

{

cout << "This is a recursive function " << ++i << endl;

message();

}

What is the problem?

**Question 18: (3 points )**

How can you tell from a class declaration that a virtual member function is pure?

**Question 19: (3 points )**

What is the difference between an instance member variable and a static member variable?

**Question 20: (2 points )**

Static member variables are declared inside the class declaration. Where are static member variables defined?

**Question 21: (8 points )**

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

Output:

#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;

}