CS 120 EXAM

Fall 2014

Dr. Byrnes

100 POINTS

NAME\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

QUESTION POINTS

1 \_\_\_\_\_\_\_\_ 20

2 \_\_\_\_\_\_\_\_ 20

3 \_\_\_\_\_\_\_\_ 20

4 \_\_\_\_\_\_\_\_ 20

5 \_\_\_\_\_\_\_\_ 20

100

Total \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Pointers (20 points)

a. (12 points)

In class we discussed some of the problems that arise when programming with pointers such as aliasing, dangling reference and garbage (memory leak). Given the following declarations and code segments, state the problem(s), if any, that might occur due to the code segments' execution.

int \*p, \*q ;

i. { ii. { iii. {

p = new int; p = new int; p = new int;

\*p = 5; \*p = 5; \*p = 5;

q = p; q = new int; q = p;

\*p = 10; q = p; delete q;

............ .............. ................

............ .............. ...............

} } }

i.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ii.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ iii.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

b. (8 points)

Show the output produced by each of the following code segments.

i. ii

int ALPHA[5] ={2,4,6,8,10}; int BETA[5] = {1,2,3,4,5};

int \*PTR; int \*PTR;

PTR = ALPHA; PTR = &BETA[4];

for(int i = 0; i < 5;i++) cout << \*PTR << endl;

cout << \*(PTR+i) << endl; PTR--;

cout << \*PTR << endl;

\*PTR = 8;

cout << BETA[3] << endl;

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

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

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

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

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

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

2. Class Basics (20 points, 5 points each)

Answer the following questions based on the class interface given below.

class JabberWocky {

public:

JabberWocky();

void FeedJabberWocky();

private:

char Sex;

string Color;

bool Hungry;

};

• Does the class supply a constructor? If so, identify the type of constructor provided.

• Identify all methods defined in the class.

• Identify all data members declared in the class.

• Show the code statements required to create a JabberWocky object named **myJW and to feed myJW**.

3. Definitions (20 points)

a. (5 points)

What is a *typedef*? Give two reasons for using type definitions (*typedefs*) within a program.

b. (5 points)

Briefly describe the semantics of *pass by value* and *pass by reference* for formal parameters in a function or method signature.

c. (5 points)

Briefly explain the purpose of separating a program solution into a *header file* and an *implementation file*.

d. (5 points)

Briefly explain the purpose of a precondition and post condition in function or method documentation.

4. Data Aggregates-Enumerations and Arrays (20 points)

a. (5 points)

Provide a C++ definition for an enumerated type called **Elements** that has the following named constants; *CHLORINE, HYDROGEN, OXYGEN, MERCURY, IRON and SULFER*.

b. (5 points)

Provide a C++ type definition for an array of 118 **Elements**. The array type should be called **Chemicals**.

c. (5 points)

Declare a C++ array called **ChemTable** of type **Chemicals** - the type defined in 2b.

d. (5 points)

Write a C++ for loop that initializes the array of 2c). The loop should ask the user to enter the chemical’s name and then place the name in the appropriate array position.

5. Numeric Data Representation/Addressing (20 points)

a. (4 points)

Convert the following decimal numbers to 8-bit 2s-complement binary numbers:

i. 125 base 10 \_ \_ \_ \_ \_ \_ \_ \_

ii. -76 base 10 \_ \_ \_ \_ \_ \_ \_ \_

b. (4 points)

Convert the following 2s-complement 8-bit numbers to base 10.

i. 11111111 \_\_\_\_\_\_\_\_\_\_\_\_

ii. 01110110 \_\_\_\_\_\_\_\_\_\_\_\_

c. (4 points)

Perform the following 8-bit 2’s complement addition and indicate whether overflow occurs.

i. 1011 1011

+1111 1110 overflow?

ii. 0011 1111

+1100 0010 overflow

d. (8 points)

Given the following "pseudo" declaration of a 2-D array:

*type* matrix[8][5];

Show the general addressing formula used to reference an item at matrix[i][j].

Formula = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

If matrix is stored at address 1000 (base 10) in memory and *type* is an integer requiring 4-bytes of storage where would m[4][2] be located in memory – (use a decimal address)?

Address = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ base 10.

For scrap/notes.