Syntax Problems:

1. Fill in the comments indicating whatever changes are occurring.

**NOTE:** I will refer to dereferenced pointers as whatever they are pointing to.

**NOTE 2:** It’s a good idea to draw out images as you read the solutions below

#include <iostream>

using namespace std;

int main()

{

const int ARRAY\_SIZE = 6;

int\* ptrA, \*ptrB, \*ptrC;

int x = 20, y = 40;

int z[ARRAY\_SIZE] = {3, 6, 9, 12, 15, 18};

ptrA = &x; // ptrA now points to the location of x

ptrC = &y; // ptrC now points to the location of y

ptrB = &z[1]; // ptrB now points to the location of z[1]

x += z[4]; // x = 20 + 15 = 35

\*ptrC = y + \*ptrA; // y = y + x = 40 + 35 = 75

\*ptrB += \*(ptrB - 1); // z[1] = z[1] + z[0] = 6 + 3 = 9

ptrB--; // ptrB now points to z[0], or ptrB = &z[0]

ptrC = ptrA; // ptrC now points to x

ptrA = ptrB + 3; // ptrA now point to z[0+3], or ptrA = &z[3]

ptrB = &y; // ptrB now points to the location of y

\*ptrA = \*ptrB / 5; // z[3] = y / 5 = 75 / 5 = 15

\*ptrB = \*ptrB - 25; // y = y - 25 = 75 - 25 = 50

\*ptrC /= 7; // x = x / 7 = 35 / 7 = 5

cout << x << endl; // 5

cout << y << endl; // 50

for(int i = 0; i < ARRAY\_SIZE; ++i)

{

cout << z[i] << endl; // prints elements of z, or: 3 9 9 15 15 18

}

return 0;

}

1. Write out how to safely create a dynamically allocated integer array given an integer pointer *cake\_list* and a user inputted integer *num\_cakes*. Assume *num\_cakes* is initialized already and *cake\_list* is not pointing to anything.

if ( num\_cakes > 0 ) // verify that the integer given

{

}

if ( NULL == cake\_list ) // if the new keyword didn’t work or num\_cakes <= 0

{

}

1. Fill in the following code using proper pointer manipulation (i.e. delete whenever you use new). **NOTE:** It is highly suggested to write pseudocode first before writing out the actual code.

// Resizes an array (arr) to new\_size, keeping its elements intact

// NOTE: If new\_size < current\_size, truncate elements

// NOTE 2: Assume new\_size >= 1

void resize\_array(double\* arr, int current\_size, int new\_size)

{

double\* new\_arr = new double[new\_size]; // allocate new memory block

if( new\_arr ) // another way to see if new\_arr points somewhere

{

// (ix < current\_size) for new\_size >= current\_size case (grow)

// (ix < new\_size) for new\_size <= current\_size case (shrink)

for(int ix = 0; (ix < current\_size) && (ix < new\_size); ++ix)

{

new\_arr[ix] = arr[ix]; // copy all possible elements in arr

}

delete arr; // “free” the memory of the original copy

arr = new\_arr; // point arr to the new array block

}

else // if new\_arr wasn’t allocated

{

cout << “ERROR: Unable to resize array due to memory constraints.”;

}

}