CS 121 - Worksheet 12 pt2 - More Pointer Stuff

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;
}
```

2. 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
{

}
```

3. 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.”;
        }
      }
```