This worksheet is for your use during and after lecture. It will not be collected or graded, but I think you will find it a useful tool as you learn C++ and study for the exams. Explain all false answers for the "True or False" questions; in general, show enough work and provide enough explanation so that this sheet is a useful pre-exam review. I will be happy to review your answers with you during office-hours, via Email, or instant messaging.

1. Write a single C++ statement that declares a(n):

   (a) `double` array with space for 10 elements.

   Solution: `double theArray[10];`

   (b) `int` array with element values -3, -4, and 100.

   Solution:
   ```
   int theArray[] = { -3, -4, 100 };
   ```

   (c) `double` array of 10,000 elements initialized to zero.

   Solution:
   ```
   double theArray[10000] = { 0 };
   ```

   (d) `char` array with 32 implicitly declared values.

   Solution:
   Take your pick:
   ```
   char theArray[33] = { 'a'};
   char theArray[34] = { 'a', 'b'};
   char theArray[35] = { 'a', 'b', 'c'};
   ...
   ```

   (e) `int` array of `SIZE` elements (assume `SIZE` is a previously declared constant integral variable).

   Solution: `int theArray[SIZE];`

   (f) `int` array with `SIZE` elements and the first three elements equal to 1, 2, and 3.

   Solution:
   ```
   int theArray[SIZE] = { 1, 2, 3 };
   ```

   (g) `bool` array with implicitly declared size and space for 7 elements.

   Solution:
   ```
   bool theArray[]={0,0,0,0,0,0,0};
   ```

2. Consider your answer to part f above. For what reason(s) might your answer generate a compiler error?

   Solution: `SIZE` may be less than 3 (worse yet, less than zero).

3. In an array declaration, there are two ways to specify the size of the array, one explicitly and one implicitly. Cite examples of both and explain what makes each implicit or explicit.

> **Solution:**
> `char ImplicitSize[] = { 'a', 'b', 'c'};` Implicit because the size of the array (3) is inferred by the number of array elements.
> `char ExplicitSize[3];` Explicit, since a 3-element array is always allocated, even though none of the element values are known.

4. In an array declaration, there are two ways to specify the value of a specific array element, one explicitly and one implicitly. Cite examples of both and explain what makes them implicit or explicit.

> **Solution:** `char ImplicitValue[2] = {'a'};` Implicit because the value of `ImplicitValue[1]` is zero, even though a literal `0` is not in the declaration. inferred by the number of array elements.
> `char ExplicitValue[1] = {'a'};` Explicit, since the ASCII code of lowercase "a" is stored specificlly into `ExplicitValue[0]`.

5. Square brackets (`[]`) mean two different things depending on their contextual use in C++ arrays. Describe the two ways that square brackets are used, cite C++ statements as examples.

> **Solution:**
> The first way is to identify a newly declared identifier as an array: `double data[3];` In this case, **if** a number exists inside of the square brackets, the value is interpretted as the **number of elements** the array should hold.
> The second way is for accessing specific elements of an array: `data[2] = data[0]*data[1];` In this case there **must** be a number inside of the square brackets, and that number is interpretted as the **element offset**.

6. Write a single C++ statement that declares:

   (a) A 3×3 2*d* array of integers.

   > **Solution:** `int theArray[3][3];`

   (b) A 2×4 2*d* array of doubles where each row element contains the value of its column *number*.

   > **Solution:** `double theArray[] = { { 1, 2, 3, 4 }, { 1, 2, 3, 4 }, };`

   (c) A 3×2 2*d* array of doubles where each column element contains its row index and the number of rows is implicitely declared.

   > **Solution:** `double theArray[][2] = { { 0, 0 }, { 1, 1 }, { 2, 2 } };`

   (d) A `SIZE`$\times$`SIZE` 2*d* array of doubles where each element is initialized to zero.

   > **Solution:** `int theArray[SIZE][SIZE] = {};`

(e) A 4×16 2*d* array of integers where the first element of each row is initialized to its row index, and all subsequent elements are initialized to zero.

Solution: `int theArray[4][16] = { {0}, {1}, {2}, {3}, };`

(f) A 3×32 2*d* array of Booleans with the number of rows implicitely declared and each element initialized to `false`.

Solution: `bool theArray[][32] = { {}, {}, {}, };`

7. Suppose a one dimensional array of integers is declared with `SIZE` elements. A user provides an initial (valid) index, a stride (either positive or negative, `stride > -SIZE`), and a count greater than zero from the keyboard. Let these variables be named `array`, `index`, `stride`, and `count`. If the array was: `{1, 2, 3, 4, 5, 6, 7, 8}` and `index=6`, `stride=1`, and `count=4` the application should calculate `7+8+1+2` (wrapping around to the beginning of the array). If `index=1`, `stride=-2`, and `count=3` the application should calculate `2+8+6` (wrapping around to the end of the array).

   Write a snippit of C++ that makes these calculations correctly, **and does not use an `if` statement**.

Solution:

```
1  #include <cstdlib>
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      const int SIZE(8);
7      const int array[SIZE] = {1,2,3,4,5,6,7,8};
8      int index, stride, count;
9      cout << "Enter␣0<=index<=7,␣stride>=-8,␣and␣count>0:␣" << flush;
10     cin >> index >> stride >> count;
11     if( index < 0 || index >7 || count <= 0 || stride < -SIZE) {
12         cout << "Invalid␣input." << endl;
13         exit(1);
14     }
15
16     int sum = 0;
17     while( count-- ) {
18         sum += array[index];
19         index = (index + (SIZE + stride)) % SIZE;
20     }
21
22     cout << "Result:␣" << sum << endl;
23     return 0;
24 }
```

8. Suppose a two dimensional `M` row by `N` column matrix is declared, and the elements are valued from `0` to `M*N-1`.

   (a) What are the indices of the `M`-th valued element?

Solution: `[M/N][M%N]`

(b) What are the indices of the `N`-th valued element?

Solution: `[N/N][N%N]` = `[1][0]`

(c) What are the indices of the element with value `x` between `0` and `M*N-1`?

Solution: `[x/N][x%N]`