CS 162 Worksheet 2

1. Scope
    a) refers to the lifetime and visibility of a variable
    b) In C/C++, a variable's scope is the {} it is declared within
    c) Main idea: A variable dies at the end of the {…} it was declared in

    Ex: For the following program, answer the following questions:
    a. Where does `temp` and `num` out of scope?

    b. Is num still accessible in `f1()`? Why or why not?

    c. Can you print `temp` outside the if? Why or why not?

```
1. void f1(); //function prototype
2. int main () {
3.     int num;
4.     cin >> num;
5.     if (num > 0) {
6.         int temp = 2 * num;
7.         cout << temp << endl;
8.     }
9.     cout << temp << endl;
10.       f1();
11.    return 0;
12.}
13.
14.void f1 () {
15.    cout << num << endl;
16.}
```

2. C++ string
    In C++, the string type (i.e., class type) is provided to handle this. To use them, include the `<string>` library. Answer the following questions regarding C++ strings:
    a) How to declare a string, named `my_str`, and initialize it with "`hello world`"?

    b) How to print `my_str` out?

    c) How to access each element in `my_str`? For example, how to change the first element to 'H'?

d) How to get the length (size) of `my_str`? How to access the very last element?

***For extended learning:
In C, strings are:

- Character arrays (i.e., `char mystring [64];`)
- Terminated with a NULL character (i.e., `'\0'`)
- Processed using C string library (i.e., `<cstring>`)

3. (Pseudo) Random number in C++
   In C++, (pseudo) random number generation is accomplished with the rand() function.
   - rand() returns an integer between 0 and RAND_MAX
   - RAND_MAX is an integer constant defined in <cstdlib>, the value is implementation defined

   Question:
   a) How could you generate a value that is either 0 or 1?

   b) What else needs to be done so that rand() would generate different random numbers in each execution?

4. Functions – primary unit of code decomposition and abstraction in C/C++
   A function has:
   - A name
   - Zero or more parameters
   - 0 (void) or 1 return

   Trace through the following program and answer the questions in comments.

```
…
int max (int, int); //function prototype

int main () {
   int x, y, mx;
   cin >> x >> y; //assume user types 5 and 7

   //state why each of the following is a bad function call:
   mx = int max (x, y);
   mx = max (int x, int y);
   mx = max (a, b);
   max (x, y);

   //put the correct function call here: (mx should store the max of x and y)
   ____________________________;
```

```
    cout << mx << endl; // what's the output?
    cout << max(0, x) << endl; // what's the output?

    return 0;
}

int max (int a, int b) {
    if (a > b)
        return a;
    return b;
}
```

5. Error handling
    a) Write design and pseudocode: prompt the user for a positive integer until a valid one is provided. The user may enter anything, including strings.
    Hint: `stoi()` from `<string>` is used to convert a string to an int

6. 1D static array
    An array is a ***fixed size***, ***named collection*** of ***ordered*** variables of the ***same type*** that are accessed with an index and stored contiguously in memory.
    – Fixed size: Cannot grow or shrink
    – Named collection: One name to refer to all variables in the array
    – Ordered / Accessed with an index: Individual element is accessed with its /index (using [])
    – Same Type: Elements in one array must all be the same type
    Ex.
    a. Declare an array of 50 doubles. Follow up: What are their initial values?

    b. What is the size and index range of the following array? How to access the last element?
    `string my_strings[10];`

    c. For the array declared in b, how to print all elements?

    d. How to pass an array into functions?

Coming From Other Languages?
SIMILARITIES: Like Python and Java, C/C++ arrays

- Use 0-based indexing (beginning element at index 0)
- loops can be used to iterate over all the elements of an array

DIFFERENCES: Unlike Python and Java, C/C++ arrays

- Are fixed size (size must be a constant) and then cannot grow easily after being declared
- Do not remember their size (no len() or .length) nor bounds-check an access (so accessing array[1048726] will happily execute in C/C++ and likely cause a crash (i.e. "Segmentation Fault")
- Are NOT objects (no .append() or .length) in C/C++