CS 31: Midterm 2 Review -- 11/16/2015

- Arrays
  - Rules for specifying size:
    - **Must** be included in the brackets
    - **Cannot** involve a variable unless it is a constant known at compile time
    - The only time size can be left out is when a list of its contents is included
  - Passing Arrays to Functions
    - Parameter Syntax
      - (..., *type* name[], ...)
    - Arrays are default passed by reference
      - Any changes made to the array will be retained outside of the function scope
- For length of:
  - strings: name.size() #include <string>
  - cstrings: strlen(name) #include <cstring>
- C Strings:
  - The end of a C string is marked by a null byte (**'\0'**)
    - Null byte has ascii value 0
    - **strlen** simply looks for the null byte for you
    - for (int i = 0; arr[i] != '\0'; i++)
    - arr[i] != **'\0'** and arr[i] != **0** are the same, as ascii value of **'\0'** is **0**
  - cin.getline(s, 50);
  - strcpy(s, t); // strcpy(destination, source);
  - strcat(s, "!!!"); ← concatenate
  - strcmp(a, b)
    - negative if a comes before b
    - 0 if a equals b
    - positive if a comes after b
- Modifiers: &, const, *, etc.
- Pass by Value
  - By default, all parameters in C++ are <u>pass by value</u>.
  - Every pass by value parameter is **copied** into the function
- Pass by Reference
  - A **reference** to a variable is passed to the function instead of a copy of the variable
  - Syntax: add an & between parameter type and name
    - int& x, bool& b, string& s
  - If these variables are **changed inside** the function, then they will also be **changed outside.**

- As the name of a type:
  - double double
  - double&reference to double
  - double* pointer to double
- In an expression:
  - &x means "generate a pointer to x" "address of x"
  - *p means "the object that p points to" "follow the pointer p" "dereference p"

- Libraries
  - #include <library>, #include <iostream>, #include <string>, #include <cctype>
  - iostream → input/output stream
- Namespaces:
  - using namespace std;
- Modifying variables
  - Integer division truncates after decimal point
  - Use double
- Strings
  - used to store blocks of text
  - strings can be initialized through literals
    - string s = “hello”
  - individual characters called by s[x]

```
string s = "Hello";
for (int k = 0; k != s.size(); k++)
        cout << s[k] << endl;
```

- cctype
  - #include <cctype>
  - Returns true/false for certain conditions
  - isalpha(x), isdigit(x), islower(x), ispunct(x), isspace(x), isupper(x), tolower(x), toupper(x)
- Ignoring characters
  - cin.ignore (in numChars, char delim)
  - cin.ignore(10000, ‘\n’);
  - Put after entering (cin) a number (int or double) and before entering a string (getline(cin, xyz))
- switch statements
  - switch(expression)
    {
    case constant expression:
        //stuff
        break; //optional
    case x:
        //stuff
        break;
    default: //optional
        //stuff
    }
- do while loop:
  - do
    { //stuff
    } while (cond.);
- for loops
  - for(init; condition; increment)
- Don’t forget to put **;** and what happens when there are negative numbers or zero
- Show how many decimal places:

```
cout.setf(ios::fixed);
cout.precision(2);
```