# W02 Task Submission (Variables and Types)

**You should watch the videos under Instructor Videos for this module before completing these tasks. You may need to perform additional research to solve these tasks. You are encouraged to use the Support Forum to get help with these tasks. After completing your tasks, please submit your original C source files. You do not need to submit files you did not write. In each of your programs, use comments to explain what the code does.**

1. Declare variables of each of the 8 simple types (char, short, int, long, long long, float, double, long double). Declare other variables of each of the 8 simple types combined with each of the 2 modifiers (signed, unsigned). Display to the user how much memory each of these types **uses**, using printf and the sizeof operator. Are any of these 24 combinations invalid? **You may simply comment out the combinations that are shown to be invalid by the compiler.** It is not required to print out the values of each variable but you may do so if you wish, as long as you initialize the variables first. **Remember that different types of variables have different format specifiers when using printf!**

2. **(Optional)** Define an enumeration using enum with three elements and display to the user the size of each one and the size of the enum itself.

3. Define a structure that contains 3 **members** of different **types. Declare** a variable of this type. Display to the user the size of each element and the total size of the structure. **You do not need to display the elements to the user but you may if you wish.**

4. Define a union that contains 3 **members** of different **types. Declare** a variable of this type. Display to the user the size of each element and the total size of the union. **You may use the same members that you used for the structure if you wish.**

5. Create your own type using typedef and struct that **contains** 3 **members** of different types. Create an array of 3 elements of this type. Ask the user to input values for one of these array elements **(all three member values)** and display the values back to the user. (When passing numeric variables to scanf, use the '&' address operator in front of the variable name to pass the address of the variable to scanf. Addresses and pointers will be covered in more detail later in the course.)

6. (Optional) Use the mathematical and assignment operators = * / % + - += -= *= /= %= using types, variables, and values of your choice to perform mathematical operations and display the results to the user. Use each operator at least once.

7. Create a variable that has a signed type. Assign a negative number to that variable. Create a second variable that has the same type but is unsigned. Using type casting, assign the signed value to the unsigned variable. Display to the user the values of both variables. Use the correct printf specifier to display the signed and unsigned types. **Remember that signed and unsigned variables have different format specifiers when using printf!**

8. Ask the user to input a floating point value and assign this to a floating point variable using scanf. Display to the user **the number they entered in three different formats: scientific, fixed point, and shortest representation (hint: these are three different specifiers to printf).** (When passing numeric variables to scanf, use the '&' address operator in front of the

variable name to pass the address of the variable to scanf. Addresses and pointers will be covered in more detail later in the course.)

9. Using 12 different printf statements (4 different groups of 3 statements each), demonstrate the difference between prefix and postfix ++ and -- operators. In each group of these statements, display three values: the value of a variable before the operation, the value of the operation, and the value of the variable after the operation. **When using printf, you can pass in an expression instead of a single variable, like this: printf("Value is %d\n", value++);**