# CS2275 Exercise 12 – Shifts, Masks, Packing and Unpacking

Module 12 objectives: shifts and other bitwise operators, masking, packing and unpacking, struts

Module 12 Readings: Lippman -4.8 page 141

Using Dev-C++, create a .cpp source file titled ex12-<your last name>.cpp that contains solutions to the following problems. As with all assignments in this course, be sure to include a block comment just about the function indicating its purpose and use a really good function name. Include code under main to test your function by allowing the user to enter the needed parameters. Make sure you test your functions sufficiently to insure correctness.

1. Write a C++ function packShorts that accepts a strut containing 4 unsigned shorts, each containing a number between 0 and 2^16-1, and returns an unsigned long long, containing the first short in bits 0-15, the second short in bits 16-31, etc. Use shifts, bitwise & and |, and masking.
2. Write a C++ function unPackShorts that accepts the unsigned long long created by your packShorts function, and returns a pointer to a strut containing 4 unsigned shorts. Use packShorts and unPackShorts to test each other.
3. Write a C++ function packUnsignedShortArray that accepts an array of unsigned shorts, along with an integer indicating its length, and returns an array of unsigned long longs that contained the packed unsigned shorts. Use your packShorts function. How long with your returned array be as compared with the unsigned shorts array? (Put the answer in the block comment for this method.)
4. Write a C++ function unPackUnsignedShortArray that accepts the unsigned long long array created by packUnsignedShortArray and returns an array of unsigned shorts. Use unPackUnsignedShortArray and packUnsignedShortArray to teach each other.

Grading: 25 points each. Style – 10: poor variable or poor function names or no block comment with a goal statement for a function. Solutions that have significantly more lines that needed will be docked points for lack of elegance.

**REMINDERS**

* code that does not compile will not be graded
* the grader should not need to modify/uncomment your code to test it. Provide a test mechanism allowing the grader to enter various tests for teach function.