# 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. Using the packRGB function in note 12.2 as a template, write a C++ function packChar that accepts a strut containing 4 chars (c0, c1, c2, and c3), each containing a number between 0 and 2^8-1, and returns an unsigned 32bit int, containing the first char in bits 0-7, the second char in bits 8-15, etc. Use shifts, bitwise & and |, and masking.
2. Write a C++ function unPackChar that accepts the unsigned int created by your packChar function, and returns a pointer to a strut containing 4 chars (c0, c1, c2, and c3). Use packChar and unPackChar to test each other.
3. Create a char pointer foo. Cast the unsigned int returned from packRGB as a char pointer and assign it to foo. What is contained in foo[0], foo[1], foo[2], and foo[3]?
4. Write a C++ function packCharArray that accepts an array of char c of length 12 and an integer indicating its length, and returns a pointer to an array packedInt of unsigned ints that contains all the characters in c. Array packedInt should be length 3. PackCharArray should call your packChar function.
5. Write a C++ function unPackUnsignedIntArray that accepts the unsigned int array packedInt created by packCharArray of length 3 and returns a pointer to an array of char c of length 12. Use your unPackUnsignedInt and packCharArray to test each other.

Grading: 20 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.