*The experienced C programmer will probably be a bit frustrated with the attention to details required by the Ada compiler. You will not have your favorite "tricks" available to fool the compiler into doing something out of the ordinary. The Ada compiler cannot be fooled. -* <http://perso.telecom-paristech.fr/~pautet/Ada95/intro.htm>

For this programming assignment, you will be writing an ada program that will do the following:

1. Create a binary array of 1’s and 0’s that is of length 16.
   1. Your binary array should be a user-defined type called BINARY\_ARRAY that can ONLY store numbers that are of type BINARY\_NUMBER
   2. BINARY\_NUMBER is another user-defined type that is a numeric value of either 0 or 1.
   3. To populate your array, you must generate 16 random 0’s or 1’s using a for loop
2. Your program must have the following FUNCTIONS:
   1. Bin\_To\_Int accepts one BINARY\_ARRAY as a parameter and returns the integer equivalent of that binary value. You must calculate this using a loop. DO NOT use a function built-in to ada to do this.
   2. Int\_To\_Bin accepts an integer value (you may want to limit the size as we only have 16 “bits” to store it in) and returns a BINARY\_ARRAY set to the binary equivalent of that integer value. The binary array should be in the correct order (least significant bit at the far right).
   3. Overloaded “+”
      1. One version of this should add two BINARY\_ARRAYs and return a BINARY\_ARRAY as a result. Don’t worry about overflows.
      2. The other version of this should accept a BINARY\_ARRAY and an integer and return a BINARY\_ARRAY as a result.
   4. Overloaded “-“ (same as above except subtract the second from the first parameter)
3. Your program must have the following procedures:
   1. Reverse\_Bin\_Arr accepts a BINARY\_ARRAY and simply reverses it.
   2. Print\_Bin\_Arr accepts a BINARY\_ARRAY and prints it to the console.
4. Your “main” should do the following:
   1. #1 described above
   2. Call your Bin\_To\_Int function from 2a and display the result on the console
   3. Call your Int\_To\_Bin function and save the return value into a second BINARY\_ARRAY.
   4. Call your Print\_Bin\_Arr procedure from 3b to display the newly created binary array.
   5. Use your overloaded + function on arrays 1 and 2, and save the return value into a third BINARY\_ARRAY. Print this as well using your print procedure
   6. Use your overloaded – function on arrays 1 and 2, and save the return value into a fourth BINARY\_ARRAY. Print this as well.
   7. Finally, reverse the fourth BINARY\_ARRAY using your Reverse\_Bin\_Arr procedure from 3a, and print the result.

TIP: You will need to create more variables than the one single array I mentioned in the instructions above.

**Helpful Links:**

**Download here. You want GNAT ada 2016**: <http://libre.adacore.com/download/>

**Old but very useful** (careful with the chapter on packages. Newer versions of ada require you to write them in two separate files. The specification belongs in .ads, and the implementation belongs in .adb):

<http://perso.telecom-paristech.fr/~pautet/Ada95/intro.htm>

**Other Useful Links**:

<http://www.cs.fsu.edu/~baker/ada/examples/>

<http://www.cristhianny.com/others/ada_tutorial_introduction_code.html>

<http://www.adapower.com/adapower1/articles/class.html>