Exercice : Décimal en binaire

```cpp
/* ========================================================================
 *
 * Decimal to binary and reverse conversions.
 *
 * Version: 1.0 (170901)
 * Author: J.-C. Chappelier (EPFL, 2017)
 *
 * ========================================================================
 */
#include <iostream>
#include <string>
using namespace std;

/* ========================================================================
 * This is the first algorithm of interest:
 * convertion from positive decimal to binary.
 */
string binary_nonnegative(int n)
{
  if (n <= 0) {
    return "0"s;
  }

  // here n >= 1
  string output;
  while (n > 0) {
    output = char('0' + n%2) + output;
    /* We could also make use of a less compact form, like:
       if (n%2 == 1) {
           output = '1' + output;
       } else {
           output = '0' + output;
       }
    */

    n /= 2;
  }

  return output;
}

/* ========================================================================
 * This is the second algorithm of interest:
 * convertion from positive binary (no sign bit) to decimal.
 */
unsigned int decimal_nonnegative(string const& binary)
// Note: from C++-17, make use of string_view rather than "string const&"
{
  unsigned int output(0);
  unsigned int power(1);

  /* There are more advanced way to iterate over a const string in C++ (crbegin(),
   * crend()). This is the very basic way for beginners in C++ programming.
   */
  if (not binary.empty()) {
    for (size_t i(binary.size() - 1); i < binary.size(); --i) {
      if (binary[i] == '1') {
        output += power;
      }
      power *= 2;
    }
  }

  return output;
}
```

```cpp
/* ============================================================================
 * This is the third algorithm of interest:
 * two's complement of binary string
 */
string twos_complement(string const& binary)
// Note: from C++-17, make use of string_view rather than "string const&"
{
  string output(binary); // starts from a copy

  if (not binary.empty()) {

    // invert all bits left to the least-signicant 1 (LS1)

    // first search for LS1
    /* There are more advanced way to iterate over a string in C++ (cbegin(),
     * cend()). This is the very basic way for beginners in C++ programming.
     */
    size_t i(output.size() - 1);
    while ((i < output.size()) and (output[i] != '1')) --i;

    // skip that LS1
    --i;

    // then invert all "higher" bits
    while (i < output.size()) {
      if (output[i] == '1') {
        output[i] = '0';
      } else {
        output[i] = '1';
      }
      --i;
    }
  }

  return output;
}

/* ============================================================================
 * This is the forth algorithm of interest:
 * convertion from (signed) decimal to binary, making use of former algorithms.
 */
string binary(int n)
{
  // Add the bit sign to the corresponding "unsigned" representation
  if (n < 0) {
    return "1"s + twos_complement(binary_nonnegative(-n));
    /* could also be written as:
     * return twos_complement("0"s + binary_nonnegative(-n));
     */
  }
  return "0"s + binary_nonnegative(n);
}

/* ============================================================================
 * This is the fifth algorithm of interest:
 * convertion from binary (with sign bit) to decimal.
 */
int decimal(string const& binary)
// Note: from C++-17, make use of string_view rather than "string const&"
{
  // test sign bit
  if (not binary.empty() and (binary[0] == '1')) {
    return -decimal_nonnegative(twos_complement(binary));
  }

  return decimal_nonnegative(binary);
}
```

```cpp
/* ============================================================================
 * All the rest below is not the core of the exercise but is just for
 * convenience.
 */

/* ============================================================================
 * Tool function: test an int value and its binary writing
 */
void test(int n)
{
  const string result(binary(n));
  cout << n << " is " << result << endl;
  cout << "and " << result << " is indeed " << decimal(result) << '.' << endl;
}

/* ============================================================================
 * Tool function: ask for some integer.
 */
int require_int()
{
  int value(0);
  cout << "Enter some integer: ";
  cin >> value;
  return value;
}

/* ============================================================================
 * Tool function: ask to continue or not.
 */
bool go_ahead() {
  char read('x');
  do {
    cout << "Shall we continue (y/n)? ";
    cin >> read;
  } while ((read != 'y') and (read != 'Y') and
           (read != 'n') and (read != 'N'));
  return (read == 'y') or (read == 'Y');
}

// ============================================================================
int main()
{
  const int t(42);
  cout << binary_nonnegative(t) << endl;
  test(t);
  test(-t);

  do {
    test(require_int());
  } while (go_ahead());

  return 0;
}
```