Enunciado:

Resuelva los siguientes ejercicios en C++14 sobre recursión básica. Los ejercicios están, más o menos, en orden de dificultad. Utilice el estándar C++14 en la solución de sus problemas. No olvide compilar con los *flags* apropiados para detectar *warnings* y errores.

1. Diseñe e implemente un programa que utilice recursión para calcular la k-ésima potencia de un número entero n, donde k es un entero no negativo.

2. Diseñe e implemente un programa que reciba un `string` y retorne la versión invertida de ese `string` (el primer caracter debe ser el último del `string` original, etc). Su solución debe ser recursiva.

3. Diseñe e implemente un programa que utilice recursión para determinar si un `string` es un palíndromo. El prototipo de su función debe ser tal que el parámetro de entrada es el `string` y retorna `true` si el `string` es palíndromo, y `false` en caso contrario. Su diseño debe manejar mayúsculas, minúsculas y casos mezclados de estas.

4. Diseñe e implemente un programa que utilice recursión para calcular

$$C(n,k) = \frac{n!}{k!(n-k)!}.$$

   No puede utilizar ciclos ni multiplicaciones ni la función recursiva `factorial(...)`. Recuerde que los términos $C(n,k)$ pueden obtenerse del triángulo de Pascal.

5. Diseñe e implemente un algoritmo recursivo para determinar todas las permutaciones de los caracteres en una cadena s. Por ejemplo, si la cadena es `s = "bus"`, su algoritmo debe encontrar `"bus"`, `"bsu"`, `"usb"`, `"ubs"`, `"sub"` y `"sbu"`. Su programa debe funcionar con cadenas con caracteres repetidos. Por ejemplo, si ingresa la cadena `"AABB"`, el programa produce solo seis permutaciones: `"AABB"`, `"ABAB"`, `"ABBA"`, `"BAAB"`, `"BABA"` y `"BBAA"`. Es decir, no se registran duplicados.

6. [*Reverse array.*] Suponga que tiene un `vector<data_type>` de longitud n y con el tipo de dato `data_type`. Usando recursión, invierta los elementos del `vector<data_type>`. La función que implementa esta solución debe tener el prototipo

```
1 void reverse_vector(vector<data_type> & vec);
```

   Es decir para un vector `{1,3,...,8,7}` se debe obtener el resultado `{7,8,...,3,1}`. Es posible que necesite una función extra (*helper function*) en la solución, con el objetivo de satisfacer el prototipo de `reverse_vector`. Si tiene problemas planteando la recursión, solucione el problema iterativamente primero.