PISTAS.

**Pista 1**

Código para recorrer el **std::vector** con un iterador **const\_iterator**. Debes entender el código y adaptarlo para recorrer también la **std::list**. Aquí te dejo un ejemplo de código:

std::vector<int> v = {1, 2, 3, 4, 5}; std::vector<int>::const\_iterator it; for (it = v.cend(); it != v.cbegin(); ) { --it; std::cout << \*it << std::endl; }

**Pista 2**

Cómo definir una función genérica que acepte cualquier contenedor que tenga definido el tipo de iterador **const\_iterator**. Aquí te dejo un ejemplo de código:

template<typename Container> void print\_reversed(const Container& cont) { typename Container::const\_iterator it; for (it = cont.cend(); it != cont.cbegin(); ) { --it; std::cout << \*it << std::endl; } }

**Pista 3**

Cómo utilizar la función **print\_reversed** para imprimir los elementos del **std::vector**. Aquí te dejo un ejemplo de código:

std::vector<int> v = {1, 2, 3, 4, 5}; print\_reversed(v);

**Pista 4**

Cómo utilizar la función **print\_reversed** para imprimir los elementos de la **std::list**. Aquí te dejo un ejemplo de código:

std::list<int> l = {10, 20, 30, 40, 50}; print\_reversed(l);

**Pista 5**

Cómo crear un alias de tipo para un iterador **const\_iterator** genérico que funcione con cualquier tipo de contenedor. Aquí te dejo un ejemplo de código:

template<typename Container> void print\_reversed(const Container& cont) { using const\_iterator = typename Container::const\_iterator; const\_iterator it; for (it = cont.cend(); it != cont.cbegin(); ) { --it; std::cout << \*it << std::endl; } }