Ejercicio 3.3: Realizar las siguientes tareas(usar stl):

- Escribir una función que reciba un número entero $n \geq 2$ retorne un vector con la descomposición en factores primos. Ejemplo: si $n = 24$, se debe retornar $x = (2, 2, 2, 3)$
- Escribir una función que reciba un vector x y un entero positivo p y retorne el número de repeticiones de p en el vector.
- Escribir un programa que reciba un vector x y retorne un vector $z \in Z^{m+1}$ donde $m = max_{i=0,\ldots,n-1}\{x_i\}$. La posición z_i debe almacenar el número de veces que $x_i = i$, para todo $i \in \{0, 1, \ldots, m\}$.
  Ejemplo: $z = (z_0 = 0, z_1 = 0, z_2 = 3, z_3 = 1)$

Ejercicio 3.6: Dados $n+1$ números naturales $c_1, \ldots, c_n$ y K. La tarea consiste en determinar si existe un subconjunto $S \subseteq \{1, \ldots, n\}$ tal que:

$$\sum_{j \in S} c_j = K$$

Por ejemplo, dado el conjunto $\{7, 3, 2, 5, 8\}$ y $K = 17$, la respuesta es **"SI"** al seleccionar el primero, tercero y quinto elemento. Computacionalmente, el conjunto de números puede

ser almacenado como un vector $c \in \mathbb{N}^n$ y la solución puede ser expresada como un vector binario $s \in \{0,1\}^n$ ($s_i = 1$ si el número natural c_i es usado en la solución, y $s_i = 0$ caso contrario). Entonces se debe cumplir que:

$$\sum_{i=1}^{n} c_i s_i = K$$

Podemos explorar el espacio de soluciones generando una familia F de subconjuntos $\{1, \ldots, n\}$ y verificando si alguno de estos subconjuntos es una solución del problema.

Realizar las siguientes tareas:

(a) Definir la clase:

- ```
  class subconjunto
  {
  private:
  vector<int> sol;
  vector<int> c;
  int K;
  public:
  subconjunto();
  subconjunto(int n, int K0);
  bool validar_sol(vector<vector<int> > As);
  //sobrecargar <<,>>.
  };
  ```
- **subconjunto():** Constructor.
- **subconjunto(int n, int K0):** Constructor. Generar n números aleatorios menores que $K0$ y almacenarlos en c.
- **validar_sol(*vector < vector < int > > As*):** Cada fila de la matriz As representa un vector solución s. Si alguna fila satisface la condición del problema, entonces dicha fila debe ser almacenada en el vector *sol* de la clase y se retorna true.

(b) Recibir una instancia del problema desde teclado o un archivo usando >>.

(c) Generar una matriz aleatoria $As \in \{0,1\}^{m \times n}$ y eliminar filas duplicadas.

(d) Usando la matriz anterior, verificar si en As existe una fila solución del problema.

(e) Presentar la instancia y la solución del problema(si existe) usando <<.
  ```

Ejercicio 3.8: Se dispone de una mochila con capacidad limitada de almacenamiento W y un conjunto de n objetos. Cada objeto tiene peso w_i y nos proporciona un beneficio c_i. El problema surge en seleccionar un subconjunto de objetos consecutivos para ingresarlos en la mochila de forma que nuestro beneficio sea máximo sin exceder la capacidad, es decir, necesitamos encontrar enteros $l \leq k$ tal que $\sum_{i=l}^{k} w_i \leq W$ y $\sum_{i=l}^{k} c_i$ sea maximizada. Para almacenar la información se propone:

- La clase objeto debe ser representada como un par de números reales donde la primera componente representa el peso y la segunda el beneficio.
- La clase mochila usará la siguiente declaración:

```
class mochila:private vector<objeto>
{
private:
double W;   int l;   int k;
public:
};
```

Completar las siguientes declaraciones:

(a) Definir los constructores de las clases.

(b) Sobrecargar los operadores <<, >>

(c) **void resolver();** Encontrar los índices l, k para la solución del problema.

(d) Leer la información desde un archivo y retornar la solución en pantalla.

Ejercicio 3.10: La conjetura de Goldbach es uno de los problemas abiertos más antiguos en matemáticas. Su enunciado menciona lo siguiente:

> **Caso fuerte:** Todo número par mayor que 2, puede ser escrito como la suma de dos números primos.
> **Caso débil:** Todo número impar mayor que 5, puede ser escrito como la suma de tres números primos

Verificar computacionalmente la conjetura usando un conjunto finito de números primos.

- Definir la clase:

```
class goldbach:public vector<int>
{
private:
int n;
int p;
int q;
int r;
public:
.....
};
```

- **goldbach():** Constructor de la clase.
- Sobrecarga del operador >>. Recibe un número entero positivo k. La función debe encontrar y almacenar en el objeto todos los primos $p_i \leq n = k$.
- *void conjetura(int m):* Recibe un entero par $5 < m \leq n$ y encuentra los primos p, q, r que validan la conjetura.
- Sobrecarga del operador <<, retornando los valores p, q, r