Enunciado:

Resuelva los siguientes ejercicios sobre variaciones de búsqueda lineal y binaria. Utilice el estándar C++14 en la solución de sus problemas. No olvide compilar con los *flags* apropiados para detectar *warnings* y errores.

1. Dado un arreglo de $n-1$ enteros, estos están en el rango de 1 a n. No hay duplicados en el arreglo así que falta uno de los enteros en la lista. Diseñe e implemente un algoritmo para encontrar el entero faltante.

2. Implemente un algoritmo para encontrar *e imprimir* el elemento más chico y el segundo más chico en un vector de cadenas de caracteres (`string`). En su implementación use el siguiente prototipo

   ```
   void two_smallest(vector<string>& arr);
   ```

3. Dado un vector de enteros que inicialmente aumenta y luego disminuye, encuentre el valor máximo en el vector. Su código solución debe tener el siguiente prototipo:

   ```
   int findMaximum(vector<int>& arr);
   ```

   donde `arr` es el vector de enteros. Existen al menos dos formas de implementar este algoritmo con diferentes tiempos de ejecución: una lineal y otra logarítmica.

4. *Identity*. Dado un arreglo `a[]` de `N` enteros distintos (positivos o negativos) en orden ascendente. Diseñe un algoritmo para encontrar un índice `i` tal que `a[i] = i`, si tal índice existe.

5. La búsqueda por interpolación es un algoritmo que puede usarse sobre arreglos que están ordenados y que preferiblemente tengan sus elementos uniformemente distribuidos (no hay acumulación de elementos en un subintervalo particular).

   En lugar de partir el arreglo a la mitad como en búsqueda binaria, el algoritmo intenta descartar mas de la mitad del arreglo al buscar una posición que posiblemente aísle el valor que se busca en la parte de menor tamaño de una partición. Para lograr esto, el punto *pos* en donde se divide un subarreglo del arreglo arr[] que comienza en la posición $left$ y termina en la posición $right$, se calcula de acuerdo a

   $$pos = left + \frac{(X - \text{arr}[left])(right - left)}{\text{arr}[right] - \text{arr}[left]}, \tag{1}$$

   en donde X representa el valor que se busca.

   Al igual que en búsqueda binaria, en cada iteración hay que calcular *pos* y verificar si se ha encontrado X en esta posición o si X puede encontrarse a la derecha o a la izquierda de la misma. El procedimiento se repite mientras que los valores del arreglo arr[$left$] y arr[$right$] sean distintos y X se encuentre entre estos.

   Implemente el algoritmo de búsqueda por interpolación y utilícelo en un programa para buscar un entero en un arreglo de enteros ordenados.

6. [*Local minimum in an array.*] Considere un `vector<int> vec` de números enteros. Diseñe un algoritmo para encontrar **un** mínimo local en `vec`. Un *mínimo local* de un vector $A = \{A_1, \ldots, A_N\}$ se define como la posición o índice $i \in [1, N]$ tal que tanto $A_{i-1} \geqslant A_i$ como $A_i \leqslant A_{i+1}$ son expresiones verdaderas. Implemente su algoritmo como la función

   ```
   int anti_peak(const vector<int> & vec);
   ```

   que toma como parámetro el vector y retorna el índice correspondiente a **un** mínimo local. Note que su algoritmo debe retornar solamente un mínimo local, cualquiera, el primero que encuentre. Note también que el vector se pasa por referencia (`&`) y el modificador `const` implica que no se puede modificar dentro de la función.