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. Observe la variación del algoritmo de búsqueda lineal o secuencial que se muestra abajo. Considere un vector de números enteros `vec` de tamaño N (o equivalentemente `vec.size()`) y una llave de búsqueda `key`.

   ```
   1  int quick_linear_search(vector<int> vec, int key) {
   2      int idx = 0;
   3      vec.push_back(key);
   4
   5      while (vec[idx] != key) idx++;
   6      vec.pop_back();
   7
   8      if (idx < vec.size()) return idx;
   9      else return -1;
   10 }
   ```

   Donde `push_back(val)` agrega el ítem `val` al final del vector y `pop_back()` lo elimina.

   Usando las reglas para el cálculo de tiempos de ejecución vistas en clase, para este algoritmo:

   - Calcule el tiempo de ejecución como un polinomio que depende de N, es decir, $T(N) = \sum_{k=0}^{z} a_k N^k$. Su tarea es, entonces, encontrar las constantes a_k.

- Compare esta variación de búsqueda lineal con el algoritmo clásico. ¿Cuál es más rápido y por qué? Justifique su respuesta usando la información obtenida en el numeral anterior.

A continuación se muestra el algoritmo clásico de búsqueda lineal:

```
int normal_linear_search(vector<int> vec, int key) {
    for (int idx = 0; idx < vec.size(); idx++) {
        if (vec[idx] == key) {
            return idx;
        }
    }
    return -1;
}
```

Recuerde justificar clara y formalmente todas sus respuestas.