Enunciado:

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

1. Implemente un mapa (usando una tabla hash) que use llaves tipo `string`, sacadas de la lista de palabras en el archivo `palabras.txt`, mientras que el valor puede ser de cualquier tipo de dato (el parámetro del `template`). El mapa usa una función hash simple para obtener la posición en la tabla y la técnica de encadenamiento separado (*separate chaining*) para resolver las colisiones. El constructor parametrizado recibe un elemento del mismo tipo de dato que el valor del mapa. Este será el valor retornado por `get` para indicar que un elemento dado no se encontró en la tabla. Para valores enteros, por ejemplo, es útil usar el máximo valor entero posible: Para esto incluya las librerías `<cstdint>` y `<limits>` y obtenga el valor máximo como `std::numeric_limits<std::int32_t>::max()`.

   La interfaz de la estructura de datos a implementar se muestra a continuación:

```
const int TABLE_SIZE = 1013; // prime number

template <typename VT>
struct KeyValueNode {
  string key;
  VT value;
  KeyValueNode<VT> *next;
};

template <typename VT>
class HashMap {
private:
  // pointer to pointers to buckets
  KeyValueNode<VT> **table;

  int tableSize; // size of the pointer table
  int count;     // number of elements in table

  // default value to return when search fails
  VT notfound;

  // search for key "key" inside the bucket at index
  // "index" of the table
  // return the element if found, or nullptr otherwise
  KeyValueNode<VT>* search_bucket(int index, string key);

  // hash function
  unsigned int hash(string key) {
    unsigned int hashVal = 0;
    for (char ch : key)
```

```
31        hashVal += ch;
32      return hashVal % tableSize;
33    }
34
35  public:
36    HashMap(VT def);
37    ~HashMap();
38
39    int size();   // return no. of elements
40    bool empty(); // true if there are no elements
41    void clear(); // delete all elements
42
43    // chained hash search: search for elem with key = key
44    VT get(string key);
45    // return true if key is in table
46    bool search(string key);
47    // chained hash insert: insert at the head of bucket
48    bool insert(string key, VT value);
49    // chained hash remove: remove element with key = key
50    void remove(string key);
51
52    // print no. of elems in each bucket to a file filename
53    void distribution(const string &filename);
54  };
```