EJERECICIOS HASH

PROBLEMA 1 – Sistema de Registro de Estudiantes Usando Hash

Planteamiento

La universidad registra estudiantes por su número de matrícula (ID).
Cada estudiante tiene un ID entero único.

Queremos almacenar **N estudiantes** en una **tabla hash** para permitir que las búsquedas sean rápidas.

Debes implementar:

- Hash con **resolución de colisiones por sondeo lineal (linear probing)**
- Insertar todos los IDs
- Luego recibir **Q búsquedas** y para cada ID consultado imprimir:
  - `"ENCONTRADO"` si está en la tabla
  - `"NO ENCONTRADO"` si no está

Razonamiento

- Creamos una tabla hash de tamaño `T` (por simplicidad `T = 2*N`).
- Insertamos cada ID usando la función:

- Insertamos cada ID usando la función:
- `h(x) = x % T`
- Si la posición está ocupada → avanzamos linealmente:
- `pos = (pos + 1) % T`
- Para buscar un ID:
  - verificamos la posición hash
  - avanzamos hasta encontrar:
    - el número buscado → éxito
    - una celda vacía → no existe

Simulación de entrada

Entrada:

```
5 3
10 22 31 4 15
22
100
4
```

Explicación:

- Insertamos: `{10,22,31,4,15}`
- Buscamos: `22`, `100`, `4`

Salida esperada:

```
ENCONTRADO
NO ENCONTRADO
ENCONTRADO
```

CODIGO

```
#include <iostream>
using namespace std;

int main() {
    int N, Q;
    cin >> N >> Q;

    int T = 2 * N;                  // tamaño de la tabla hash
    int* table = new int[T];
    bool* used = new bool[T];       // indica si la celda tiene algún valor

    for (int i = 0; i < T; i++) used[i] = false;

    // INSERTAR ESTUDIANTES
    for (int i = 0; i < N; i++) {
        int id;
        cin >> id;

        int pos = id % T;
        if (pos < 0) pos += T;

        while (used[pos]) {
            pos = (pos + 1) % T;     // sondeo lineal
        }

        table[pos] = id;
        used[pos] = true;
    }

    // BUSQUEDAS
    for (int i = 0; i < Q; i++) {
        int x;
        cin >> x;

        int pos = x % T;
        if (pos < 0) pos += T;

        bool found = false;
        int steps = 0;

        while (used[pos] && steps < T) {
            if (table[pos] == x) {
```

```
            found = true;
            break;
        }
        pos = (pos + 1) % T;
        steps++;
    }

    if (found) cout << "ENCONTRADO\n";
    else cout << "NO ENCONTRADO\n";
  }

  delete[] table;
  delete[] used;

  return 0;
}
```

PROBLEMA 2 – Contador de Palabras con Hash (muy explicable)

Planteamiento

En un libro se quieren contar cuántas veces aparece cada palabra.

Dado un texto con **N palabras**, queremos registrar cada palabra en una tabla hash y contar sus ocurrencias.

Luego se piden **Q consultas**, donde cada consulta es una palabra, y se debe imprimir cuántas veces aparece.

Razonamiento

- Usamos una tabla hash con:
  - **arreglo dinámico de cadenas** (`string*`)
  - **arreglo de contadores** (`int*`)
- Hash:

```
h(s) = suma de códigos ASCII de s % T
```

- Si hay colisión:
  - usamos sondeo lineal
- Para cada palabra:
  - si existe → aumentamos contador
  - si no existe → la insertamos con contador 1
- Para buscar:
  - se aplica sondeo lineal hasta hallarla o encontrar celda vacía

Simulación de entrada

Entrada:

```
7 3
hola mundo hola como estas hola mundo
hola
mundo
adios
```

Salida esperada:

```
3
2
0
```

CODIGO

```
#include <iostream>
#include <string>
using namespace std;

// Funcion hash para strings
int hashString(const string& s, int T) {
    int sum = 0;
    for (char c : s) sum += (int)c;
    return sum % T;
}

int main() {
    int N, Q;
    cin >> N >> Q;

    int T = 2 * N + 5;     // tamaño tabla
    string* table = new string[T];
    int* count = new int[T];
    bool* used = new bool[T];

    for (int i = 0; i < T; i++) {
        used[i] = false;
        count[i] = 0;
    }

    // INSERTAR PALABRAS
    for (int i = 0; i < N; i++) {
        string w;
        cin >> w;

        int pos = hashString(w, T);

        while (used[pos] && table[pos] != w) {
            pos = (pos + 1) % T;
        }

        if (!used[pos]) {
            table[pos] = w;
            used[pos] = true;
            count[pos] = 1;
        } else {
            count[pos]++;
```

```
        }
    }

    // CONSULTAS
    for (int i = 0; i < Q; i++) {
        string w;
        cin >> w;

        int pos = hashString(w, T);
        int steps = 0;
        bool found = false;

        while (used[pos] && steps < T) {
            if (table[pos] == w) {
                cout << count[pos] << "\n";
                found = true;
                break;
            }
            pos = (pos + 1) % T;
            steps++;
        }

        if (!found) cout << "0\n";
    }

    delete[] table;
    delete[] used;
    delete[] count;

    return 0;
}
```