Project: Periodic Table - Element Information System

Overview:

This project provides a structured representation of chemical elements, encapsulating detailed information about each element in the periodic table. The program defines a `struct` to store various attributes of an element and initializes a list of elements with pre-defined data such as atomic number, symbol, name, atomic weight, electron configuration, and other key chemical and physical properties.

---

1. Project Files:

- **`main.c`**: Contains the primary code including data initialization and functions for querying and displaying element information.

---

2. Data Structure:

The `Element` structure holds information for each element of the periodic table. The structure contains various fields that describe the chemical and physical properties of the element.

Struct: `Element`

```
c
Copy code
typedef struct {
    int atomicNumber;                   // Atomic number of the element
    char symbol[3];                     // Symbol of the element (1 or 2 characters)
    char name[50];                      // Full name of the element
    double atomicWeight;                // Atomic weight of the element
    char category[50];                  // Element category (e.g., "Non-metal",
"Transition metal")
    int group;                          // Group number in the periodic table
    int period;                         // Period number in the periodic table
    char electronConfiguration[100];    // Electron configuration of the element
    double density;                     // Density of the element in g/cm^3
    double meltingPoint;                // Melting point in Celsius
    double boilingPoint;                // Boiling point in Celsius
    double electronegativity;           // Electronegativity according to Pauling
scale
    int yearDiscovered;                 // Year when the element was discovered
    char discoverer[100];               // Name of the discoverer
} Element;
```

---

3. Element Information:

The `elements[]` array stores the details of various elements in the periodic table, using the `Element` structure. Each element is described by:

- **Atomic Number**: Unique number of protons in the nucleus.
- **Symbol**: Short representation of the element.
- **Name**: Full name of the element.

- **Atomic Weight**: Average atomic mass.
- **Category**: Type of element (e.g., Alkali metal, Non-metal).
- **Group and Period**: Location in the periodic table.
- **Electron Configuration**: Electron distribution across atomic orbitals.
- **Density**: Mass per unit volume (g/cm^3).
- **Melting and Boiling Points**: Temperature points at which the element changes state.
- **Electronegativity**: Tendency to attract electrons in a bond (Pauling scale).
- **Year Discovered and Discoverer**: Historical information about the element.

Example Initialization:

```
c
Copy code
Element elements[] = {
    {1, "H", "Hydrogen", 1.008, "Non-metal", 1, 1, "1s1", 0.08988, -259.16, -252.87,
2.20, 1766, "Cavendish"},
    {2, "He", "Helium", 4.0026, "Noble gas", 18, 1, "1s2", 0.1786, -272.20, -268.93, -
1.00, 1895, "Ramsay"},
    ...
    {36, "Kr", "Krypton", 83.798, "Noble gas", 18, 4, "1s2 2s2 2p6 3s2 3p6 4s2 3d10
4p6", 0.003733, -157.37, -153.22, 3.00, 1898, "Ramsay"}
};
```

---

4. Functions:

The project can include various functions to interact with the elements data, such as:

- **`searchByAtomicNumber(int atomicNumber)`**:
  - Finds an element by its atomic number and displays its details.

```
c
Copy code
void searchByAtomicNumber(int atomicNumber) {
    for (int i = 0; i < sizeof(elements) / sizeof(Element); i++) {
        if (elements[i].atomicNumber == atomicNumber) {
            // Code to display element details
        }
    }
}
```

- **`searchByName(char* name)`**:
  - Finds an element by its name and displays its details.

```
c
Copy code
void searchByName(char* name) {
    for (int i = 0; i < sizeof(elements) / sizeof(Element); i++) {
        if (strcmp(elements[i].name, name) == 0) {
            // Code to display element details
        }
    }
}
```

- **`displayAllElements()`**:
  - Displays a list of all elements in the periodic table.

```
c
Copy code
void displayAllElements() {
    for (int i = 0; i < sizeof(elements) / sizeof(Element); i++) {
```

```
            printf("%s (%d) - %s\n", elements[i].name, elements[i].atomicNumber,
    elements[i].category);
        }
    }
```

---

5. Example Output:

Sample output when searching for an element by atomic number or name:

```
yaml
Copy code
Atomic Number: 6
Name: Carbon
Symbol: C
Atomic Weight: 12.011
Category: Non-metal
Group: 14
Period: 2
Electron Configuration: 1s2 2s2 2p2
Density: 2.267 g/cm^3
Melting Point: 3550°C
Boiling Point: 4027°C
Electronegativity: 2.55
Year Discovered: 1789
Discoverer: Lavoisier
```

---

6. Features:

- **Query Elements**: Users can search for elements based on atomic number or name.
- **Display All Elements**: A complete list of the elements in the periodic table can be printed with key information.
- **Element Properties**: Detailed information about the physical, chemical, and historical properties of each element.
- **Expandable Data**: Additional elements or data fields can be easily added.

---

7. Future Enhancements:

- **Interactive User Interface**: Add a user-friendly command-line or graphical interface for better interaction.
- **Element Classification Search**: Search elements based on their category (e.g., metals, non-metals).
- **Sorting and Filtering**: Implement sorting by properties like atomic weight, electronegativity, etc.
- **Data Import/Export**: Allow import and export of element data from external files.
- **Periodic Table Visualization**: Add visual representation of the periodic table.

---

9. References:

- **Periodic Table Data**: Information on atomic numbers, weights, and other properties obtained from standard chemical databases.