Modern Periodic Table

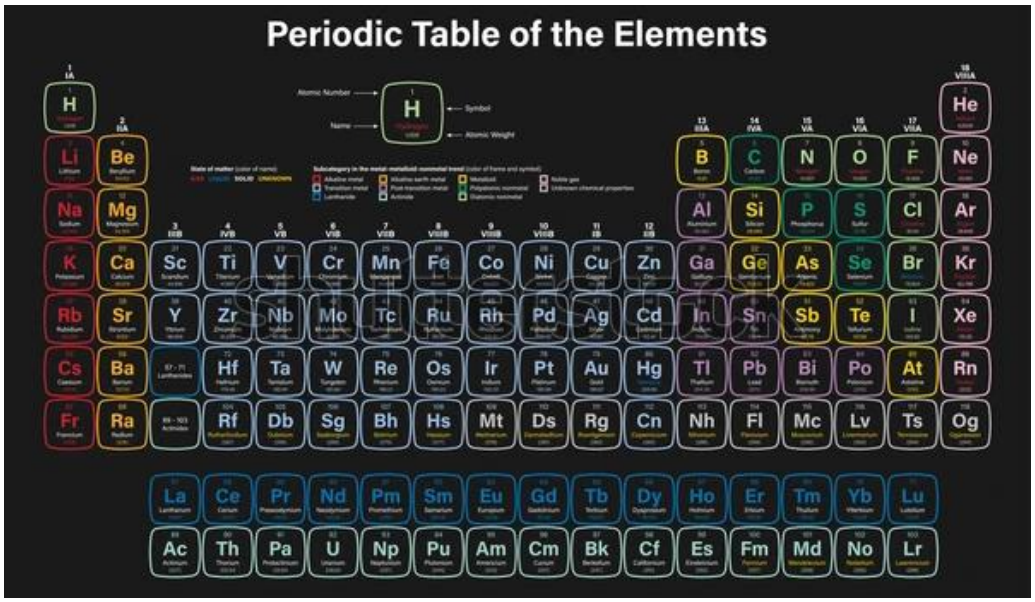
Description:

The periodic table, also known as the periodic table of elements, is a tabular display of the chemical elements, which are arranged by atomic number, electron configuration, and recurring chemical properties. The structure of the table shows periodic trends. The seven rows of the table, called periods, generally have metals on the left and nonmetals on the right. The columns, called groups, contain elements with similar chemical behaviours. Six groups have accepted names as well as assigned numbers: for example, group 17 elements are the halogens; and group 18 are the noble gases. Also displayed are four simple rectangular areas or blocks associated with the filling of different atomic orbitals.

The elements from atomic numbers 1 (hydrogen) to 118 (oganesson) have all been discovered or synthesized, completing seven full rows of the periodic table. The first 94 elements, hydrogen to plutonium, all occur naturally, though some are found only in trace amounts and a few were discovered in nature only after having first been synthesized. Elements 95 to 118 have only been synthesized in laboratories, nuclear reactors, or nuclear explosions. The synthesis of elements having higher atomic numbers is currently being pursued: these elements would begin an eighth row, and theoretical work has been done to suggest possible candidates for this extension. Numerous synthetic radioisotopes of naturally occurring elements have also been produced in laboratories.

The organization of the periodic table can be used to derive relationships between the various element properties, and also to predict chemical properties and behaviours of undiscovered or newly synthesized elements. Russian chemist Dmitri Mendeleev published the first recognizable periodic table in 1869, developed mainly to illustrate periodic trends of the then-known elements. He also predicted some properties of unidentified elements that were expected to fill gaps within the table. Most of his forecasts soon proved to be correct, culminating with the discovery of gallium and germanium in 1875 and 1886 respectively, which corroborated his predictions Mendeleev's idea has been slowly expanded and refined with the discovery or synthesis of further new elements and the development of new theoretical models to explain chemical behaviour. The modern periodic table now provides a useful framework for analyzing chemical reactions, and continues to be widely used in chemistry, nuclear physics and other sciences.

Modern Periodic Table



Sets of elements

In chronological order, this section discusses metals and nonmetals (and metalloids); categories of elements; groups and periods; and periodic table blocks. While the recognition of metals as solid, fusible and generally malleable substances dates from antiquity, Antoine Lavoisier may have the first to formally distinguish between metals and nonmetals *('non-métalliques')* in 1789 with the publication of his 'revolutionary'[1] *Elementary Treatise on Chemistry*. In 1811, Berzelius referred to nonmetallic elements as metalloids, in reference to their ability to form oxyanions. In 1825, in a revised German edition of his *Textbook of Chemistry*, he subdivided the metalloids into three classes. These were: constantly gaseous 'gazolyta' (hydrogen, nitrogen, oxygen); real metalloids (sulfur, phosphorus, carbon, boron, silicon); and salt-forming 'halogenia' (fluorine, chlorine, bromine, iodine). Only recently, since the mid-20th century, has the term metalloid been widely used to refer to elements with intermediate or borderline properties between metals and nonmetals. Mendeleev published his periodic table in 1869, along with references to groups of families of elements, and rows or periods of his periodic table. At the same time, Hinrichs wrote that simple lines could be drawn on a periodic table in order to delimit properties of interest, such as elements having metallic lustre (in contrast to those not having such lustre). Charles Janet, in 1928, appears to have been the first to refer to the periodic table's blocks.

According to their shared physical and chemical properties, the elements can be classified into the major categories of metals, metalloids and nonmetals. Metals are generally shiny, highly conducting solids that form alloys with one another and salt-like ionic compounds with nonmetals (other than noble gases). A majority of nonmetals are colored or colorless insulating gases; nonmetals that form compounds with other nonmetals feature covalent bonding. In between metals and nonmetals are metalloids, which have intermediate or mixed properties.

Metal and nonmetals can be further classified into subcategories that show a gradation from metallic to non-metallic properties, when going left to right in the

rows. The metals may be subdivided into the highly reactive alkali metals, through the less reactive alkaline earth metals, lanthanides and actinides, via the archetypal transition metals, and ending in the physically and chemically weak post-transition metals. Nonmetals may be simply subdivided into the polyatomic nonmetals, being nearer to the metalloids and show some incipient metallic character; the essentially nonmetallic diatomic nonmetals, nonmetallic and the almost completely inert, monatomic noble gases. Specialized groupings such as refractory metals and noble metals, are examples of subsets of transition metals, also known and occasionally denoted.

Groups

A group or family is a vertical column in the periodic table. Groups usually have more significant periodic trends than periods and blocks, explained below. Modern quantum mechanical theories of atomic structure explain group trends by proposing that elements within the same group generally have the same electron configurations in their valence shell. Consequently, elements in the same group tend to have a shared chemistry and exhibit a clear trend in properties with increasing atomic number.[43] In some parts of the periodic table, such as the d-block and the f-block, horizontal similarities can be as important as, or more pronounced than, vertical similarities.

Under an international naming convention, the groups are numbered numerically from 1 to 18 from the leftmost column (the alkali metals) to the rightmost column (the noble gases).] Previously, they were known by roman numerals. In America, the roman numerals were followed by either an "A" if the group was in the s- or p-block, or a "B" if the group was in the d-block. The roman numerals used correspond to the last digit of today's naming convention (e.g. the group 4 elements were group IVB, and the group 14 elements were group IVA). In Europe, the lettering was similar, except that "A" was used if the group was before group 10, and "B" was used for groups including and after group 10. In addition, groups 8, 9 and 10 used to be treated as one triple-sized group, known collectively in both notations as group VIII. In 1988, the new IUPAC naming system was put into use, and the old group names were deprecated.

Periods

A period is a horizontal row in the periodic table. Although groups generally have more significant periodic trends, there are regions where horizontal trends are more significant than vertical group trends, such as the f-block, where the lanthanides and actinides form two substantial horizontal series of elements.

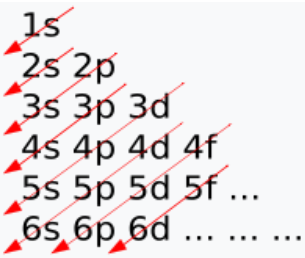
Elements in the same period show trends in atomic radius, ionization energy, electron affinity, and electronegativity. Moving left to right across a period, atomic radius usually decreases. This occurs because each successive element has an added proton and electron, which causes the electron to be drawn closer to the nucleus. This decrease in atomic radius also causes the ionization energy to increase when moving from left to right across a period. The more tightly bound an element is, the more energy is required to remove an electron. Electronegativity increases in the same manner as ionization energy because of the pull exerted on the electrons

by the nucleus. Electron affinity also shows a slight trend across a period. Metals (left side of a period) generally have a lower electron affinity than nonmetals (right side of a period), with the exception of the noble gases.

Blocks

Specific regions of the periodic table can be referred to as blocks in recognition of the sequence in which the electron shells of the elements are filled. Elements are assigned to blocks by what orbitals their valence electrons or vacancies lie in. The s-block comprises the first two groups (alkali metals and alkaline earth metals) as well as hydrogen and helium. The p-block comprises the last six groups, which are groups 13 to 18 in IUPAC group numbering (3A to 8A in American group numbering) and contains, among other elements, all of the metalloids. The d-block comprises groups 3 to 12 (or 3B to 2B in American group numbering) and contains all of the transition metals. The f-block, often offset below the rest of the periodic table, has no group numbers and comprises most of the lanthanides and actinides. A hypothetical g-block is expected to begin around element 121, a few elements away from what is currently known.

Electron configuration



The electron configuration or organisation of electrons orbiting neutral atoms shows a recurring pattern or periodicity. The electrons occupy a series of electron shells (numbered 1, 2, and so on). Each shell consists of one or more subshells (named s, p, d, f and g). As atomic number increases, electrons progressively fill these shells and subshells more or less according to the Madelung rule or energy ordering rule, as shown in the diagram. The electron configuration for neon, for example, is $1s^2\ 2s^2\ 2p^6$. With an atomic number of ten, neon has two electrons in the first shell, and eight electrons in the second shell; there are two electrons in the s subshell and six in the p subshell. In periodic table terms, the first time an electron occupies a new shell corresponds to the start of each new period, these positions being occupied by hydrogen and the alkali metals.

Since the properties of an element are mostly determined by its electron configuration, the properties of the elements likewise show recurring patterns or periodic behaviour, some examples of which are shown in the diagrams below for atomic radii, ionization energy and electron affinity. It is this periodicity of properties, manifestations of which were noticed well before the underlying theory was developed, that led to the establishment of the periodic law (the properties of the elements recur at varying intervals) and the formulation of the first periodic

tables. The periodic law may then be successively clarified as: depending on atomic weight; depending on atomic number; and depending on the total number of s, p, d, and f electrons in each atom. The cycles last 2, 6, 10, and 14 elements respectively.

There is additionally an internal "double periodicity" that splits the shells in half; this arises because the first half of the electrons going into a particular type of subshell fill unoccupied orbitals, but the second half have to fill already occupied orbitals, following Hund's rule of maximum multiplicity. The second half thus suffer additional repulsion that causes the trend to split between first-half and second-half elements; this is for example evident when observing the ionisation energies of the 2p elements, in which the triads B-C-N and O-F-Ne show increases, but oxygen actually has a first ionisation slightly lower than that of nitrogen as it is easier to remove the extra, paired electron.

Atomic radii

Atomic radii vary in a predictable and explainable manner across the periodic table. For instance, the radii generally decrease along each period of the table, from the alkali metals to the noble gases; and increase down each group. The radius increases sharply between the noble gas at the end of each period and the alkali metal at the beginning of the next period. These trends of the atomic radii (and of various other chemical and physical properties of the elements) can be explained by the electron shell theory of the atom; they provided important evidence for the development and confirmation of quantum theory.

The electrons in the 4f-subshell, which is progressively filled from lanthanum (element 57) to ytterbium (element 70), are not particularly effective at shielding the increasing nuclear charge from the sub-shells further out. The elements immediately following the lanthanides have atomic radii that are smaller than would be expected and that are almost identical to the atomic radii of the elements immediately above them. Hence lutetium has virtually the same atomic radius (and chemistry) as yttrium, hafnium has virtually the same atomic radius (and chemistry) as zirconium, and tantalum has an atomic radius similar to niobium, and so forth. This is an effect of the lanthanide contraction: a similar actinide contraction also exists. The effect of the lanthanide contraction is noticeable up to platinum (element 78), after which it is masked by a relativistic effect known as the inert pair effect.[62] The d-block contraction, which is a similar effect between the d-block and p-block, is less pronounced than the lanthanide contraction but arises from a similar cause.

Such contractions exist throughout the table, but are chemically most relevant for the lanthanides with their almost constant +3 oxidation state.

Ionization energy

The first ionization energy is the energy it takes to remove one electron from an atom, the second ionization energy is the energy it takes to remove a second electron from the atom, and so on. For a given atom, successive ionization energies increase with the degree of ionization. For magnesium as an example, the first

ionization energy is 738 kJ/mol and the second is 1450 kJ/mol. Electrons in the closer orbitals experience greater forces of electrostatic attraction; thus, their removal requires increasingly more energy. Ionization energy becomes greater up and to the right of the periodic table.

Large jumps in the successive molar ionization energies occur when removing an electron from a noble gas (complete electron shell) configuration. For magnesium again, the first two molar ionization energies of magnesium given above correspond to removing the two 3s electrons, and the third ionization energy is a much larger 7730 kJ/mol, for the removal of a 2p electron from the very stable neon-like configuration of Mg^{2+}. Similar jumps occur in the ionization energies of other third-row atoms.

Electronegativity

Electronegativity is the tendency of an atom to attract a shared pair of electrons. An atom's electronegativity is affected by both its atomic number and the distance between the valence electrons and the nucleus. The higher its electronegativity, the more an element attracts electrons. It was first proposed by Linus Pauling in 1932. In general, electronegativity increases on passing from left to right along a period, and decreases on descending a group. Hence, fluorine is the most electronegative of the elements, while caesium is the least, at least of those elements for which substantial data is available.

There are some exceptions to this general rule. Gallium and germanium have higher electronegativities than aluminium and silicon respectively because of the d-block contraction. Elements of the fourth period immediately after the first row of the transition metals have unusually small atomic radii because the 3d-electrons are not effective at shielding the increased nuclear charge, and smaller atomic size correlates with higher electronegativity. The anomalously high electronegativity of lead, particularly when compared to thallium and bismuth, is an artifact of electronegativity varying with oxidation state: its electronegativity conforms better to trends if it is quoted for the +2 state instead of the +4 state

Oxidation number

With some minor exceptions, oxidation numbers among the elements show four main trends according to their periodic table geographic location: left; middle; right; and south. On the left (groups 1 to 4, not including the f-block elements, and also niobium, tantalum, and probably dubnium in group 5), the highest most stable oxidation number is the group number, with lower oxidation states being less stable. In the middle (groups 3 to 11), higher oxidation states become more stable going down each group. Group 12 is an exception to this trend; they behave as if they were located on the left side of the table. On the right, higher oxidation states tend to become less stable going down a group. The shift between these trends is continuous: for example, group 3 also has lower oxidation states most stable in its lightest member (scandium, with $CsScCl_3$ for example known in the +2 state),[76] and group 12 is predicted to have copernicium more readily showing oxidation states above +2.

Modern Periodic Table

The lanthanides positioned along the south of the table are distinguished by having the +3 oxidation state in common; this is their most stable state. The early actinides show a pattern of oxidation states somewhat similar to those of their period 6 and 7 transition metal congeners; the later actinides are more similar to the lanthanides, though the last ones (excluding lawrencium) have an increasingly important +2 oxidation state that becomes the most stable state for nobelium.

Algorithm

Linear Search (Array A, Value x)

Step 1: Set i to 1
Step 2: if i > n then go to step 7
Step 3: if A[i] = x then go to step 6
Step 4: Set i to i + 1
Step 5: Go to Step 2
Step 6: Print Element x Found at index i and go to step 8
Step 7: Print element not found
Step 8: Exit

Pseudocode

```
procedure linear_search (list, value)

   for each item in the list
      if match item == value
         return the item's location
      end if
   end for

end procedure
```

Linear Search-

- Linear Search is the simplest searching algorithm.
- It traverses the array sequentially to locate the required element.
- It searches for an element by comparing it with each element of the array one by one.
- So, it is also called as Sequential Search.

Linear Search Algorithm is applied when-

- No information is given about the array.
- The given array is unsorted or the elements are unordered.
- The list of data items is smaller.

Modern Periodic Table

Code:

```
#include<iostream>
#include<conio.h>
#include<string.h>
#include<stdio.h>
#include<windows.h>
#include<process.h>

using namespace std;

//Initializing structures.
struct element{
                short atomicno;
                char name[50];
                char ec[50];
                short group;
                short period;
                char block;
                float atomicw;
                float en;
                short atomicr;
                char origin[100];
                float meltp;
                float boilp;
                float density;
              }elem[115];
int main()
{
   int n,t,a,i;
   char name[50],c;
   //Displaying introduction.
   system("COLOR 8F");
   cout<<"\n\n\n\n\n\n\n\n\n\n\n\t\tA PROGRAM BY SAHIL GOLIWAR, RISHABH JAIN
AND MOHAMMADAKRAM SHEIKH";
   Sleep(3000);
   system("cls");


   //Displaying the modern periodic table.
   cout<<"\n\t\t\tTHE MODERN PERIODIC TABLE\n";
   cout<<"\t\t\t------------------------\n";
   cout<<"\tH                                                 He"<<"\n\n";
   cout<<"\tLi Be                                B  C   N  O   F  Ne"<<"\n\n";
   cout<<"\tNa Mg                                Al Si  P  S   Cl Ar"<<"\n\n";
   cout<<"\tK  Ca Sc Ti V  Cr Mn Fe Co Ni Cu  Zn  Ga Ge  As Se  Br Kr"<<"\n\n";
   cout<<"\tRb Sr Y  Zr Nb Mo Tc Ru Rh Pd Ag  Cd  In Sn  Sb Te  I  Xe"<<"\n\n";
```

```
    cout<<"\tCs Ba La Hf Ta W  Re Os Ir Pt Au  Hg  Tl Pb  Bi Po  At Rn"<<"\n\n";
    cout<<"\tFr Ra Ac Rf Db Sg Bh Hs Mt Ds Uuu Uub -  Uuq -  Uuh"<<"\n\n\n";
    cout<<"\t\t Ce Pr Nd Pm Sm Eu Gd Tb Dy Ho Er Tm Yb Lu"<<"\n\n";
    cout<<"\t\t Th Pa U  Np Pu Am Cm Bk Cf Es Fm Md No Lr"<<"\n\n\n";
    system("PAUSE");
    system("cls");
    //Initializing atomic numbers.
    for(i=1;i<=114;++i)
    {
                elem[i].atomicno=i;
    }
    //Initializing element names and electronic configurations.
    {
                strcpy(elem[1].name,"Hydrogen- H");         strcpy(elem[1].ec,"1s1");
                strcpy(elem[2].name,"Helium- He");          strcpy(elem[2].ec,"1s2");
                strcpy(elem[3].name,"Lithium- Li");         strcpy(elem[3].ec,"[He]
2s1");
                strcpy(elem[4].name,"Beryllium- Be");       strcpy(elem[4].ec,"[He]
2s2     ");
                strcpy(elem[5].name,"Boron- B");            strcpy(elem[5].ec,"[He] 2s2
2p1");
                strcpy(elem[6].name,"Carbon- C");           strcpy(elem[6].ec,"[He] 2s2
2p2");
                strcpy(elem[7].name,"Nitrogen- N");         strcpy(elem[7].ec,"[He]
2s2 2p3");
                strcpy(elem[8].name,"Oxygen- O");           strcpy(elem[8].ec,"[He]
2s2 2p4");
                strcpy(elem[9].name,"Fluorine- F");         strcpy(elem[9].ec,"[He] 2s2
2p5");
                strcpy(elem[10].name,"Neon- Ne");           strcpy(elem[10].ec,"[He]
2s2 2p6");
                strcpy(elem[11].name,"Sodium- Na");         strcpy(elem[11].ec,"[Ne]
3s1");
                strcpy(elem[12].name,"Magnesium- Mg");
strcpy(elem[12].ec,"[Ne] 3s2");
                strcpy(elem[13].name,"Aluminium- Al");      strcpy(elem[13].ec,"[Ne]
3s2 3p1");
                strcpy(elem[14].name,"Silicon- Si");        strcpy(elem[14].ec,"[Ne]
3s2 3p2");
                strcpy(elem[15].name,"Phosphorus- P");      strcpy(elem[15].ec,"[Ne]
3s2 3p3");
                strcpy(elem[16].name,"Sulfur- S");          strcpy(elem[16].ec,"[Ne]
3s2 3p4");
                strcpy(elem[17].name,"Chlorine- Cl");       strcpy(elem[17].ec,"[Ne]
3s2 3p5");
                strcpy(elem[18].name,"Argon- Ar");          strcpy(elem[18].ec,"[Ne]
3s2 3p6");
```

```
                strcpy(elem[19].name,"Potassium- K");       strcpy(elem[19].ec,"[Ar]
4s1");
                strcpy(elem[20].name,"Calcium- Ca");        strcpy(elem[20].ec,"[Ar]
4s2");
                strcpy(elem[21].name,"Scandium- Sc");       strcpy(elem[21].ec,"[Ar]
3d1 4s2");
                strcpy(elem[22].name,"Titanium- Ti");       strcpy(elem[22].ec,"[Ar]
3d2 4s2");
                strcpy(elem[23].name,"Vanadium- V");        strcpy(elem[23].ec,"[Ar]
3d3 4s2");
                strcpy(elem[24].name,"Chromium- Cr");       strcpy(elem[24].ec,"[Ar]
3d5 4s1");
                strcpy(elem[25].name,"Manganese- Mn");      strcpy(elem[25].ec,"[Ar]
3d5 4s2");
                strcpy(elem[26].name,"Iron- Fe");           strcpy(elem[26].ec,"[Ar]
3d6 4s2");
                strcpy(elem[27].name,"Cobalt- Co");         strcpy(elem[27].ec,"[Ar]
3d7 4s2");
                strcpy(elem[28].name,"Nickel- Ni");         strcpy(elem[28].ec,"[Ar]
3d8 4s2");
                strcpy(elem[29].name,"Copper- Cu");         strcpy(elem[29].ec,"[Ar]
3d10 4s1");
                strcpy(elem[30].name,"Zinc- Zn");           strcpy(elem[30].ec,"[Ar]
3d10 4s2");
                strcpy(elem[31].name,"Gallium- Ga");        strcpy(elem[31].ec,"[Ar]
3d10 4s2 4p1");
                strcpy(elem[32].name,"Germanium- Ge");
strcpy(elem[32].ec,"[Ar] 3d10 4s2 4p2");
                strcpy(elem[33].name,"Arsenic- As");        strcpy(elem[33].ec,"[Ar]
3d10 4s2 4p3");
                strcpy(elem[34].name,"Selenium- Se");       strcpy(elem[34].ec,"[Ar]
3d10 4s2 4p4");
                strcpy(elem[35].name,"Bromine- Br");        strcpy(elem[35].ec,"[Ar]
3d10 4s2 4p5 ");
                strcpy(elem[36].name,"Krypton- Kr");        strcpy(elem[36].ec,"[Ar]
3d10 4s2 4p6 ");
                strcpy(elem[37].name,"Rubidium- Rb");       strcpy(elem[37].ec,"[Kr]
5s1");
                strcpy(elem[38].name,"Strontium- Sr");      strcpy(elem[38].ec,"[Kr]
5s2");
                strcpy(elem[39].name,"Yttrium- Y");         strcpy(elem[39].ec,"[Kr]
4d1 5s2");
                strcpy(elem[40].name,"Zirconium- Zr");      strcpy(elem[40].ec,"[Kr]
4d2 5s2");
                strcpy(elem[41].name,"Niobium- Nb");        strcpy(elem[41].ec,"[Kr]
4d4 5s1");
```

```
                strcpy(elem[42].name,"Molybdenum- Mo");
strcpy(elem[42].ec,"[Kr] 4d5 5s1");
                strcpy(elem[43].name,"Technetium- Tc");      strcpy(elem[43].ec,"[Kr]
4d5 5s2");
                strcpy(elem[44].name,"Ruthenium- Ru");       strcpy(elem[44].ec,"[Kr]
4d7 5s1");
                strcpy(elem[45].name,"Rhodium- Rh");         strcpy(elem[45].ec,"[Kr]
4d8 5s1");
                strcpy(elem[46].name,"Palladium- Pd");       strcpy(elem[46].ec,"[Kr]
4d10");
                strcpy(elem[47].name,"Silver- Ag");          strcpy(elem[47].ec,"[Kr]
4d10 5s1");
                strcpy(elem[48].name,"Cadmium- Cd");         strcpy(elem[48].ec,"[Kr]
4d10 5s2");
                strcpy(elem[49].name,"Indium- In");          strcpy(elem[49].ec,"[Kr]
4d10 5s2 5p1");
                strcpy(elem[50].name,"Tin- Sn");             strcpy(elem[50].ec,"[Kr]
4d10 5s2 5p2");
                strcpy(elem[51].name,"Antimony- Sb");        strcpy(elem[51].ec,"[Kr]
4d10 5s2 5p3");
                strcpy(elem[52].name,"Tellurium- Te");       strcpy(elem[52].ec,"[Kr]
4d10 5s2 5p4");
                strcpy(elem[53].name,"Iodine- I");           strcpy(elem[53].ec,"[Kr]
4d10 5s2 5p5");
                strcpy(elem[54].name,"Xenon- Xe");           strcpy(elem[54].ec,"[Kr]
4d10 5s2 5p6");
                strcpy(elem[55].name,"Caesium- Cs");         strcpy(elem[55].ec,"[Xe]
6s1");
                strcpy(elem[56].name,"Barium- Ba");          strcpy(elem[56].ec,"[Xe]
6s2");
                strcpy(elem[57].name,"Lanthanum- La");       strcpy(elem[57].ec,"[Xe]
5d1 6s2");
                strcpy(elem[58].name,"Cerium- Ce");          strcpy(elem[58].ec,"[Xe]
4f1 5d1 6s2");
                strcpy(elem[59].name,"Praseodymium- Pr");
strcpy(elem[59].ec,"[Xe] 4f3 6s2");
                strcpy(elem[60].name,"Neodymium- Nd");
strcpy(elem[60].ec,"[Xe] 4f4 6s2");
                strcpy(elem[61].name,"Promethium- Pm");
strcpy(elem[61].ec,"[Xe] 4f5 6s2");
                strcpy(elem[62].name,"Samarium- Sm");        strcpy(elem[62].ec,"[Xe]
4f6 6s2");
                strcpy(elem[63].name,"Europium- Eu");        strcpy(elem[63].ec,"[Xe]
4f7 6s2");
                strcpy(elem[64].name,"Gadolinium- Gd");      strcpy(elem[64].ec,"[Xe]
4f7 5d1 6s2");
```

```
                strcpy(elem[65].name,"Terbium- Tb");        strcpy(elem[65].ec,"[Xe]
4f9 6s2");
                strcpy(elem[66].name,"Dysprosium- Dy");     strcpy(elem[66].ec,"[Xe]
4f10 6s2");
                strcpy(elem[67].name,"Holmium- Ho");        strcpy(elem[67].ec,"[Xe]
4f11 6s2");
                strcpy(elem[68].name,"Erbium- Er");         strcpy(elem[68].ec,"[Xe]
4f12 6s2");
                strcpy(elem[69].name,"Thulium- Tm");        strcpy(elem[69].ec,"[Xe]
4f13 6s2");
                strcpy(elem[70].name,"Ytterbium- Yb");      strcpy(elem[70].ec,"[Xe]
4f14 6s2");
                strcpy(elem[71].name,"Lutetium- Lu");       strcpy(elem[71].ec,"[Xe]
4f14 5d1 6s2");
                strcpy(elem[72].name,"Hafnium- Hf");        strcpy(elem[72].ec,"[Xe]
4f14 5d2 6s2");
                strcpy(elem[73].name,"Tantalum- Ta");       strcpy(elem[73].ec,"[Xe]
4f14 5d3 6s2");
                strcpy(elem[74].name,"Tungsten- W");        strcpy(elem[74].ec,"[Xe]
4f14 5d4 6s2");
                strcpy(elem[75].name,"Rhenium- Re");        strcpy(elem[75].ec,"[Xe]
4f14 5d5 6s2");
                strcpy(elem[76].name,"Osmium- Os");         strcpy(elem[76].ec,"[Xe]
4f14 5d6 6s2");
                strcpy(elem[77].name,"Iridium- Ir");        strcpy(elem[77].ec,"[Xe]
4f14 5d7 6s2");
                strcpy(elem[78].name,"Platinum- Pt");       strcpy(elem[78].ec,"[Xe]
4f14 5d9 6s1");
                strcpy(elem[79].name,"Gold- Au");           strcpy(elem[79].ec,"[Xe]
4f14 5d10 6s1");
                strcpy(elem[80].name,"Mercury- Hg");        strcpy(elem[80].ec,"[Xe]
4f14 5d10 6s2         ");
                strcpy(elem[81].name,"Thallium- Tl");       strcpy(elem[81].ec,"[Xe]
4f14 5d10 6s2 6p1");
                strcpy(elem[82].name,"Lead- Pb");           strcpy(elem[82].ec,"[Xe]
4f14 5d10 6s2 6p2");
                strcpy(elem[83].name,"Bismuth- Bi");        strcpy(elem[83].ec,"[Xe]
4f14 5d10 6s2 6p3");
                strcpy(elem[84].name,"Polonium- Po");       strcpy(elem[84].ec,"[Xe]
4f14 5d10 6s2 6p4");
                strcpy(elem[85].name,"Astatine- At");       strcpy(elem[85].ec,"[Xe]
4f14 5d10 6s2 6p5");
                strcpy(elem[86].name,"Radon- Rn");          strcpy(elem[86].ec,"[Xe]
4f14 5d10 6s2 6p6");
                strcpy(elem[87].name,"Francium- Fr");       strcpy(elem[87].ec,"[Rn]
7s1");
```

```
                strcpy(elem[88].name,"Radium- Ra");          strcpy(elem[88].ec,"[Rn]
7s2");
                strcpy(elem[89].name,"Actinium- Ac");        strcpy(elem[89].ec,"[Rn]
6d1 7s2");
                strcpy(elem[90].name,"Thorium- Th");         strcpy(elem[90].ec,"[Rn]
6d2 7s2");
                strcpy(elem[91].name,"Protactinium- Pa");    strcpy(elem[91].ec,"[Rn]
5f2 6d1 7s2");
                strcpy(elem[92].name,"Uranium- U");          strcpy(elem[92].ec,"[Rn]
5f3 6d1 7s2");
                strcpy(elem[93].name,"Neptunium- Np");
strcpy(elem[93].ec,"[Rn] 5f4 6d1 7s2");
                strcpy(elem[94].name,"Plutonium- Pu");       strcpy(elem[94].ec,"[Rn]
5f6 7s2");
                strcpy(elem[95].name,"Americium- Am");
strcpy(elem[95].ec,"[Rn] 5f7 7s2");
                strcpy(elem[96].name,"Curium- Cm");          strcpy(elem[96].ec,"[Rn]
5f7 6d1 7s2");
                strcpy(elem[97].name,"Berkelium- Bk");       strcpy(elem[97].ec,"[Rn]
5f9 7s2");
                strcpy(elem[98].name,"Californium- Cf");     strcpy(elem[98].ec,"[Rn]
5f10 7s2");
                strcpy(elem[99].name,"Einsteinium- Es");     strcpy(elem[99].ec,"[Rn]
5f11 7s2");
                strcpy(elem[100].name,"Fermium- Fm");
strcpy(elem[100].ec,"[Rn] 5f12 7s2");
                strcpy(elem[101].name,"Mendelevium- Md");
strcpy(elem[101].ec,"[Rn] 5f13 7s2");
                strcpy(elem[102].name,"Nobelium- No");
strcpy(elem[102].ec,"[Rn] 5f14 7s2");
                strcpy(elem[103].name,"Lawrencium- Lr");
strcpy(elem[103].ec,"[Rn] 5f14 7s2 7p1");
                strcpy(elem[104].name,"Rutherfordium- Rf");
strcpy(elem[104].ec,"[Rn] 5f14 6d2 7s2");
                strcpy(elem[105].name,"Dubnium- Db");
strcpy(elem[105].ec,"[Rn] 5f14 6d3 7s2  ");
                strcpy(elem[106].name,"Seaborgium- Sg");
strcpy(elem[106].ec,"[Rn] 5f14 6d4 7s2");
                strcpy(elem[107].name,"Bohrium- Bh");
strcpy(elem[107].ec,"[Rn] 5f14 6d5 7s2");
                strcpy(elem[108].name,"Hassium- Hs");        strcpy(elem[108].ec,"[Rn]
5f14 6d6 7s2");
                strcpy(elem[109].name,"Meitnerium- Mt");
strcpy(elem[109].ec,"[Rn] 5f14 6d7 7s2");
                strcpy(elem[110].name,"Darmstadtium- Ds");
strcpy(elem[110].ec,"[Rn] 5f14 6d8 7s2");
```

```
                strcpy(elem[111].name,"Roentgenium- Rg");
strcpy(elem[111].ec,"[Rn] 5f14 6d9 7s2");
                strcpy(elem[112].name,"Copernicium- Cn");
strcpy(elem[112].ec,"[Rn] 5f14 6d10 7s2");
                strcpy(elem[113].name,"Ununtrium- Uut");
strcpy(elem[113].ec,"[Rn] 5f14 6d10 7s2 7p1");
                strcpy(elem[114].name,"Flerovium- Fl");      strcpy(elem[114].ec,"[Rn]
5f14 6d10 7s2 7p2");
   }
   //Initializing atomic weight, atomic radius and electronegativity.
   {
                elem[1].atomicw=1.008;          elem[1].atomicr=53;
elem[1].en=2.20;
                elem[2].atomicw=4.002602;       elem[2].atomicr=31;
elem[2].en=0;
                elem[3].atomicw=6.94;           elem[3].atomicr=167   ;
elem[3].en=0.98;
                elem[4].atomicw=9.012182;       elem[4].atomicr=112;
elem[4].en=1.57;
                elem[5].atomicw=10.81;          elem[5].atomicr=87;
elem[5].en=2.04;
                elem[6].atomicw=12.011;         elem[6].atomicr=67;
elem[6].en=2.55;
                elem[7].atomicw=14.007;         elem[7].atomicr=56;
elem[7].en=3.04;
                elem[8].atomicw=15.999;         elem[8].atomicr=48;
elem[8].en=3.44;
                elem[9].atomicw=18.9984032;     elem[9].atomicr=42;
elem[9].en=3.98;
                elem[10].atomicw=20.1797;       elem[10].atomicr=38;
elem[10].en=0;
                elem[11].atomicw=22.98976928;   elem[11].atomicr=190;
elem[11].en=0.93;
                elem[12].atomicw=24.305;        elem[12].atomicr=145;
elem[12].en=1.31;
                elem[13].atomicw=26.9815386;    elem[13].atomicr=118;
elem[13].en=1.61;
                elem[14].atomicw=28.085;        elem[14].atomicr=111;
elem[14].en=1.90;
                elem[15].atomicw=30.973762;     elem[15].atomicr=98;
elem[15].en=2.19;
                elem[16].atomicw=32.06;         elem[16].atomicr=88;
elem[16].en=2.58;
                elem[17].atomicw=35.45;         elem[17].atomicr=79;
elem[17].en=3.16;
                elem[18].atomicw=39.948;        elem[18].atomicr=71;
elem[18].en=0;
```

```
elem[19].atomicw=39.0983;       elem[19].atomicr=243;
elem[19].en=0.82;
elem[20].atomicw=40.078;        elem[20].atomicr=194;
elem[20].en=1.00;
elem[21].atomicw=44.955912;     elem[21].atomicr=184;
elem[21].en=1.36;
elem[22].atomicw=47.867;        elem[22].atomicr=176;
elem[22].en=1.54;
elem[23].atomicw=50.9415;       elem[23].atomicr=171;
elem[23].en=1.63;
elem[24].atomicw=51.9961;       elem[24].atomicr=166;
elem[24].en=1.66;
elem[25].atomicw=54.938045;     elem[25].atomicr=161;
elem[25].en=1.55;
elem[26].atomicw=55.845;        elem[26].atomicr=156;
elem[26].en=1.83;
elem[27].atomicw=58.933195;     elem[27].atomicr=152;
elem[27].en=1.88;
elem[28].atomicw=58.6934;       elem[28].atomicr=149;
elem[28].en=1.91;
elem[29].atomicw=63.546;        elem[29].atomicr=145;
elem[29].en=1.90;
elem[30].atomicw=65.38;         elem[30].atomicr=142;
elem[30].en=1.65;
elem[31].atomicw=69.723;        elem[31].atomicr=136;
elem[31].en=1.81;
elem[32].atomicw=72.630;        elem[32].atomicr=125;
elem[32].en=2.01;
elem[33].atomicw=74.92160;      elem[33].atomicr=114;
elem[33].en=2.18;
elem[34].atomicw=78.96;         elem[34].atomicr=103;
elem[34].en=2.55;
elem[35].atomicw=79.904;        elem[35].atomicr=94;
elem[35].en=2.96;
elem[36].atomicw=83.798;        elem[36].atomicr=88;
elem[36].en=3.00;
elem[37].atomicw=85.4678;       elem[37].atomicr=265;
elem[37].en=0.82;
elem[38].atomicw=87.62;         elem[38].atomicr=219;
elem[38].en=0.95;
elem[39].atomicw=88.90585;      elem[39].atomicr=212;
elem[39].en=1.22;
elem[40].atomicw=91.224;        elem[40].atomicr=206;
elem[40].en=1.33;
elem[41].atomicw=92.90638;      elem[41].atomicr=198;
elem[41].en=1.6;
```

```
elem[42].atomicw=95.96;          elem[42].atomicr=190;
elem[42].en=2.16;
elem[43].atomicw=98;             elem[43].atomicr=183;
elem[43].en=1.9;
elem[44].atomicw=101.07;         elem[44].atomicr=178;
elem[44].en=2.2;
elem[45].atomicw=102.90550;      elem[45].atomicr=173;
elem[45].en=2.28;
elem[46].atomicw=106.42;         elem[46].atomicr=169;
elem[46].en=2.20;
elem[47].atomicw=107.8682;       elem[47].atomicr=165;
elem[47].en=1.93;
elem[48].atomicw=112.411;        elem[48].atomicr=161;
elem[48].en=1.69;
elem[49].atomicw=114.818;        elem[49].atomicr=156;
elem[49].en=1.78;
elem[50].atomicw=118.710;        elem[50].atomicr=145;
elem[50].en=1.96;
elem[51].atomicw=121.760;        elem[51].atomicr=133;
elem[51].en=2.05;
elem[52].atomicw=127.60;         elem[52].atomicr=123;
elem[52].en=2.1;
elem[53].atomicw=126.90447;      elem[53].atomicr=115;
elem[53].en=2.66;
elem[54].atomicw=131.293;        elem[54].atomicr=108;
elem[54].en=2.6;
elem[55].atomicw=132.9054519;    elem[55].atomicr=298;
elem[55].en=0.79;
elem[56].atomicw=137.327;        elem[56].atomicr=253;
elem[56].en=0.89;
elem[57].atomicw=138.90547;      elem[57].atomicr=169;
elem[57].en=1.10;
elem[58].atomicw=140.90765;      elem[58].atomicr=131;
elem[58].en=1.12;
elem[59].atomicw=140.90765;      elem[59].atomicr=247;
elem[59].en=1.13;
elem[60].atomicw=144.242;        elem[60].atomicr=206;
elem[60].en=1.14;
elem[61].atomicw=145;            elem[61].atomicr=205;
elem[61].en=0;
elem[62].atomicw=150.36;         elem[62].atomicr=238;
elem[62].en=1.17;
elem[63].atomicw=151.964;        elem[63].atomicr=231;
elem[63].en=0;
elem[64].atomicw=157.25;         elem[64].atomicr=233;
elem[64].en=1.20;
```

```
elem[65].atomicw=158.92535;      elem[65].atomicr=225;
elem[65].en=0;
elem[66].atomicw=162.500;        elem[66].atomicr=228;
elem[66].en=1.22;
elem[67].atomicw=164.93032;      elem[67].atomicr=0;
elem[67].en=1.23;
elem[68].atomicw=167.259;        elem[68].atomicr=226;
elem[68].en=1.24;
elem[69].atomicw=168.93421;      elem[69].atomicr=222;
elem[69].en=1.25;
elem[70].atomicw=173.054;        elem[70].atomicr=222;
elem[70].en=0;
elem[71].atomicw=174.9668;       elem[71].atomicr=217;
elem[71].en=1.27;
elem[72].atomicw=178.49;         elem[72].atomicr=208;
elem[72].en=1.3;
elem[73].atomicw=180.94788;      elem[73].atomicr=200;
elem[73].en=1.5;
elem[74].atomicw=183.84;         elem[74].atomicr=193;
elem[74].en=2.36;
elem[75].atomicw=186.207;        elem[75].atomicr=188;
elem[75].en=1.9;
elem[76].atomicw=190.23;         elem[76].atomicr=185;
elem[76].en=2.2;
elem[77].atomicw=192.217;        elem[77].atomicr=180;
elem[77].en=2.20;
elem[78].atomicw=195.084;        elem[78].atomicr=177;
elem[78].en=2.28;
elem[79].atomicw=196.966569;     elem[79].atomicr=174;
elem[79].en=2.54;
elem[80].atomicw=200.592;        elem[80].atomicr=171;
elem[80].en=2.00;
elem[81].atomicw=204.38;         elem[81].atomicr=156;
elem[81].en=1.62;
elem[82].atomicw=207.2;          elem[82].atomicr=154;
elem[82].en=2.33;
elem[83].atomicw=208.98040;      elem[83].atomicr=143;
elem[83].en=2.02;
elem[84].atomicw=209;            elem[84].atomicr=135;
elem[84].en=2.0;
elem[85].atomicw=210;            elem[85].atomicr=0 ;
elem[85].en=2.2;
elem[86].atomicw=222;            elem[86].atomicr=120;
elem[86].en=0;
elem[87].atomicw=223;            elem[87].atomicr=0;
elem[87].en=0.7;
```

```
elem[88].atomicw=226;          elem[88].atomicr=0;
elem[88].en=0.9;
elem[89].atomicw=227;          elem[89].atomicr=0;
elem[89].en=1.1;
elem[90].atomicw=232.03806;    elem[90].atomicr=0;
elem[90].en=1.3;
elem[91].atomicw=231.03588;    elem[91].atomicr=0;
elem[91].en=1.5;
elem[92].atomicw=238.02891;    elem[92].atomicr=0;
elem[92].en=1.38;
elem[93].atomicw=237;          elem[93].atomicr=0;
elem[93].en=1.36   ;
elem[94].atomicw=244;          elem[94].atomicr=0;
elem[94].en=1.28;
elem[95].atomicw=243;          elem[95].atomicr=0;
elem[95].en=1.3;
elem[96].atomicw=247;          elem[96].atomicr=0;
elem[96].en=1.3;
elem[97].atomicw=247;          elem[97].atomicr=0;
elem[97].en=1.3;
elem[98].atomicw=251;          elem[98].atomicr=0;
elem[98].en=1.3;
elem[99].atomicw=252;          elem[99].atomicr=0;
elem[99].en=1.3;
elem[100].atomicw=257;         elem[100].atomicr=0;
elem[100].en=1.3;
elem[101].atomicw=258;         elem[101].atomicr=0;
elem[101].en=1.3;
elem[102].atomicw=259;         elem[102].atomicr=0;
elem[102].en=1.3;
elem[103].atomicw=262;         elem[103].atomicr=0;
elem[103].en=1.3;
elem[104].atomicw=267;         elem[104].atomicr=0;
elem[104].en=1.3;
elem[105].atomicw=268;         elem[105].atomicr=0;
elem[105].en=1.3;
elem[106].atomicw=269;         elem[106].atomicr=0;
elem[106].en=0;
elem[107].atomicw=270;         elem[107].atomicr=0;
elem[107].en=0;
elem[108].atomicw=269;         elem[108].atomicr=0;
elem[108].en=0;
elem[109].atomicw=278;         elem[109].atomicr=0;
elem[109].en=0;
elem[110].atomicw=281;         elem[110].atomicr=0;
elem[110].en=0;
```

```
                elem[111].atomicw=281;          elem[111].atomicr=0;
elem[111].en=0;
                elem[112].atomicw=285;          elem[112].atomicr=0;
elem[112].en=0;
                elem[113].atomicw=286;          elem[113].atomicr=0;
elem[113].en=0;
                elem[114].atomicw=289;          elem[114].atomicr=0;
elem[114].en=0;
   }
   //Initializing melting point , boiling point and density.
   {
                elem[1].meltp=14.01;           elem[1].boilp=20.28;
elem[1].density=0.00008988;
                elem[2].meltp=0.95;           elem[2].boilp=4.22;
elem[2].density=0.0001785;
                elem[3].meltp=453.69;          elem[3].boilp=1560;
elem[3].density=0.534;
                elem[4].meltp=1560;           elem[4].boilp=2742;
elem[4].density=1.85;
                elem[5].meltp=2349;           elem[5].boilp=4200;
elem[5].density=2.34;
                elem[6].meltp=3800;           elem[6].boilp=4300;
elem[6].density=2.267;
                elem[7].meltp=63.15;          elem[7].boilp=77.36;
elem[7].density=0.0012506;
                elem[8].meltp=54.36;          elem[8].boilp=90.20;
elem[8].density=0.001429;
                elem[9].meltp=53.53;          elem[9].boilp=85.03;
elem[9].density=0.001696;
                elem[10].meltp=24.56;          elem[10].boilp=27.07;
elem[10].density=0.0008999;
                elem[11].meltp=370.87;         elem[11].boilp=1156;
elem[11].density=0.971;
                elem[12].meltp=923;           elem[12].boilp=1363;
elem[12].density=1.738;
                elem[13].meltp=933.47;         elem[13].boilp=2792;
elem[13].density=2.698;
                elem[14].meltp=1687;          elem[14].boilp=3538;
elem[14].density=2.3296;
                elem[15].meltp=317.30;         elem[15].boilp=550;
elem[15].density=1.82;
                elem[16].meltp=388.36;         elem[16].boilp=717.87;
elem[16].density=2.067;
                elem[17].meltp=171.6;          elem[17].boilp=239.11;
elem[17].density=0.003214;
                elem[18].meltp=83.80;          elem[18].boilp=87.30;
elem[18].density=0.0017837;
```

```
elem[19].meltp=336.53;  elem[19].boilp=1032;
elem[19].density=0.862;
elem[20].meltp=1115;  elem[20].boilp=1757;
elem[20].density=1.54;
elem[21].meltp=1814;  elem[21].boilp=3109;
elem[21].density=2.989;
elem[22].meltp=1941;  elem[22].boilp=3560;
elem[22].density=4.54;
elem[23].meltp=2183;  elem[23].boilp=3680;
elem[23].density=6.11;
elem[24].meltp=2180;  elem[24].boilp=2944;
elem[24].density=7.15;
elem[25].meltp=1519;  elem[25].boilp=2334;
elem[25].density=7.44;
elem[26].meltp=1811;  elem[26].boilp=3134;
elem[26].density=7.874;
elem[27].meltp=1768;  elem[27].boilp=3200;
elem[27].density=8.86;
elem[28].meltp=1728;  elem[28].boilp=3186;
elem[28].density=8.912;
elem[29].meltp=1357.77;  elem[29].boilp=2835;
elem[29].density=8.96;
elem[30].meltp=692.88;  elem[30].boilp=1180;
elem[30].density=7.134;
elem[31].meltp=302.9146;  elem[31].boilp=2477;
elem[31].density=5.907;
elem[32].meltp=1211.40;  elem[32].boilp=3106;
elem[32].density=5.323;
elem[33].meltp=1090;  elem[33].boilp=887;
elem[33].density=5.776;
elem[34].meltp=453;  elem[34].boilp=958;
elem[34].density=4.809;
elem[35].meltp=265.8;  elem[35].boilp=332.0;
elem[35].density=3.122;
elem[36].meltp=115.79;  elem[36].boilp=119.93;
elem[36].density=0.003733;
elem[37].meltp=312.46;  elem[37].boilp=961;
elem[37].density=1.532;
elem[38].meltp=1050;  elem[38].boilp=1655;
elem[38].density=2.64;
elem[39].meltp=1799;  elem[39].boilp=3609;
elem[39].density=4.469;
elem[40].meltp=2128;  elem[40].boilp=4682;
elem[40].density=6.506;
elem[41].meltp=2750;  elem[41].boilp=5017;
elem[41].density=8.57;
```

```
                elem[42].meltp=2896;          elem[42].boilp=4912;
elem[42].density=10.22;
                elem[43].meltp=2430;          elem[43].boilp=4538;
elem[43].density=11.5;
                elem[44].meltp=2607;          elem[44].boilp=4423;
elem[44].density=12.37;
                elem[45].meltp=2237;          elem[45].boilp=3968;
elem[45].density=12.41;
                elem[46].meltp=1828.05;        elem[46].boilp=3236;
elem[46].density=12.02;
                elem[47].meltp=1234.93;        elem[47].boilp=2435;
elem[47].density=10.501;
                elem[48].meltp=594.22;        elem[48].boilp=1040;
elem[48].density=8.69;
                elem[49].meltp=429.75;        elem[49].boilp=2345;
elem[49].density=7.31;
                elem[50].meltp=505.08;        elem[50].boilp=2875;
elem[50].density=7.287;
                elem[51].meltp=903.78;        elem[51].boilp=1860;
elem[51].density=6.685;
                elem[52].meltp=722.66;        elem[52].boilp=1261;
elem[52].density=6.232;
                elem[53].meltp=386.85;        elem[53].boilp=457.4;
elem[53].density=4.93;
                elem[54].meltp=161.4;         elem[54].boilp=165.03;
elem[54].density=0.005887;
                elem[55].meltp=301.59;        elem[55].boilp=944;
elem[55].density=1.873;
                elem[56].meltp=1000;          elem[56].boilp=2170;
elem[56].density=3.594;
                elem[57].meltp=1193;          elem[57].boilp=3737;
elem[57].density=6.145;
                elem[58].meltp=1068;          elem[58].boilp=3716;
elem[58].density=6.77;
                elem[59].meltp=1208;          elem[59].boilp=3793;
elem[59].density=6.773;
                elem[60].meltp=1297;          elem[60].boilp=3347;
elem[60].density=7.007;
                elem[61].meltp=1315;          elem[61].boilp=3273;
elem[61].density=7.26;
                elem[62].meltp=1345;          elem[62].boilp=2067;
elem[62].density=7.52;
                elem[63].meltp=1099;          elem[63].boilp=1802;
elem[63].density=5.243;
                elem[64].meltp=1585;          elem[64].boilp=3546;
elem[64].density=7.895;
```

```
elem[65].meltp=1629;          elem[65].boilp=3503;
elem[65].density=8.229;
elem[66].meltp=1680;          elem[66].boilp=2840;
elem[66].density=8.55;
elem[67].meltp=1734;          elem[67].boilp=2993;
elem[67].density=8.795;
elem[68].meltp=1802;          elem[68].boilp=3141;
elem[68].density=9.066;
elem[69].meltp=1818;          elem[69].boilp=2223;
elem[69].density=9.321;
elem[70].meltp=1097;          elem[70].boilp=1469;
elem[70].density=6.965;
elem[71].meltp=1925;          elem[71].boilp=3675;
elem[71].density=9.84;
elem[72].meltp=2506;          elem[72].boilp=4876;
elem[72].density=13.31;
elem[73].meltp=3290;          elem[73].boilp=5731;
elem[73].density=16.654;
elem[74].meltp=3695;          elem[74].boilp=5828;
elem[74].density=19.25;
elem[75].meltp=3459;          elem[75].boilp=5869;
elem[75].density=21.02;
elem[76].meltp=3306;          elem[76].boilp=5285;
elem[76].density=22.61;
elem[77].meltp=2719;          elem[77].boilp=4701;
elem[77].density=22.56;
elem[78].meltp=2041.4;        elem[78].boilp=4098;
elem[78].density=21.46;
elem[79].meltp=1337.33;       elem[79].boilp=3129;
elem[79].density=19.282;
elem[80].meltp=234.43;        elem[80].boilp=629.88;
elem[80].density=13.5336;
elem[81].meltp=577;           elem[81].boilp=1746;
elem[81].density=11.85;
elem[82].meltp=600.61;        elem[82].boilp=2022;
elem[82].density=11.342;
elem[83].meltp=544.7;         elem[83].boilp=1837;
elem[83].density=9.807;
elem[84].meltp=527;           elem[84].boilp=1235;
elem[84].density=9.32;
elem[85].meltp=575;           elem[85].boilp=610;
elem[85].density=7;
elem[86].meltp=202;           elem[86].boilp=211.3;
elem[86].density=0.00973;
elem[87].meltp=300;           elem[87].boilp=950;
elem[87].density=1.87;
```

```
elem[88].meltp=973;          elem[88].boilp=2010;
elem[88].density=5.5;
elem[89].meltp=1323;         elem[89].boilp=3471;
elem[89].density=10.07;
elem[90].meltp=2115;         elem[90].boilp=5061;
elem[90].density=11.72;
elem[91].meltp=1841;         elem[91].boilp=4300;
elem[91].density=15.37;
elem[92].meltp=1405.3;       elem[92].boilp=4404;
elem[92].density=18.95;
elem[93].meltp=917;          elem[93].boilp=4273;
elem[93].density=20.45;
elem[94].meltp=912.5;        elem[94].boilp=3501;
elem[94].density=19.84;
elem[95].meltp=1449;         elem[95].boilp=2880;
elem[95].density=13.69;
elem[96].meltp=1613;         elem[96].boilp=3383;
elem[96].density=13.51;
elem[97].meltp=1259;         elem[97].boilp=2900;
elem[97].density=14.79;
elem[98].meltp=1173;         elem[98].boilp=1743;
elem[98].density=15.1;
elem[99].meltp=1133;         elem[99].boilp=1269;
elem[99].density=8.84;
elem[100].meltp=1125;        elem[100].boilp=0;
elem[100].density=0;
elem[101].meltp=1100;        elem[101].boilp=0;
elem[101].density=0;
elem[102].meltp=1100;        elem[102].boilp=0;
elem[102].density=0;
elem[103].meltp=1900;        elem[103].boilp=0;
elem[103].density=0;
elem[104].meltp=2400;        elem[104].boilp=5800;
elem[104].density=23.2;
elem[105].meltp=0;           elem[105].boilp=0;
elem[105].density=29.3;
elem[106].meltp=0;           elem[106].boilp=0;
elem[106].density=35.0;
elem[107].meltp=0;           elem[107].boilp=0;
elem[107].density=37.1;
elem[108].meltp=0;           elem[108].boilp=0;
elem[108].density=40.7;
elem[109].meltp=0;           elem[109].boilp=0;
elem[109].density=37.4;
elem[110].meltp=0;           elem[110].boilp=0;
elem[110].density=34.8;
```

```
            elem[111].meltp=0;          elem[111].boilp=0;
elem[111].density=28.7;
            elem[112].meltp=0;          elem[112].boilp=357 ;
elem[112].density=23.7;
            elem[113].meltp=700;         elem[113].boilp=1400;
elem[113].density=16;
            elem[114].meltp=340;         elem[114].boilp=420;
elem[114].density=14;
   }

   //Initializing group number and block.
   {
            for(i=1;i<=114;++i)
            {
                        if(i==1 || i==3 || i==11 || i==19 || i==37 || i==55 ||
i==87)
                        {
                            elem[i].group=1;
                            elem[i].block='s';
                        }
                        else if(i==4 || i==12 || i==20 || i==38 || i==56 || i==88)
                        {
                            elem[i].group=2;
                            elem[i].block='s';
                        }
                        else if(i==21 || i==39 || i==57 || i==89)
                        {
                            elem[i].group=3;
                            elem[i].block='d';
                        }
                        else if(i==22 || i==40 || i==72 || i==104)
                        {
                            elem[i].group=4;
                            elem[i].block='d';
                        }
                        else if(i==23 || i==41 || i==73 || i==105)
                        {
                            elem[i].group=5;
                            elem[i].block='d';
                        }
                        else if(i==24 || i==42 || i==74 || i==106)
                        {
                            elem[i].group=6;
                            elem[i].block='d';
                        }
                        else if(i==25 || i==43 || i==75 || i==107)
                        {
```

```
        elem[i].group=7;
        elem[i].block='d';
    }
    else if(i==26 || i==44 || i==76 || i==108)
    {
        elem[i].group=8;
        elem[i].block='d';
    }
    else if(i==27 || i==45 || i==77 || i==109)
    {
        elem[i].group=9;
        elem[i].block='d';
    }
    else if(i==28 || i==46 || i==78 || i==110)
    {
        elem[i].group=10;
        elem[i].block='d';
    }
    else if(i==29 || i==47 || i==79 || i==111)
    {
        elem[i].group=11;
        elem[i].block='d';
    }
    else if(i==30 || i==48 || i==80 || i==112)
    {
        elem[i].group=12;
        elem[i].block='d';
    }
    else if(i==5 || i==13 || i==31 || i==49 || i==81 || i==113)
    {
        elem[i].group=13;
        elem[i].block='p';
    }
    else if(i==6 || i==14 || i==32 || i==50 || i==82 || i==114)
    {
        elem[i].group=14;
        elem[i].block='p';
    }
    else if(i==7 || i==15 || i==33 || i==51 || i==83 || i==115)
    {
        elem[i].group=15;
        elem[i].block='p';
    }
    else if(i==8 || i==16 || i==34 || i==52 || i==84 || i==116)
    {
        elem[i].group=16;
        elem[i].block='p';
```

```
                        }
                        else if(i==9 || i==17 || i==35 || i==53 || i==85 || i==117)
                        {
                            elem[i].group=17;
                            elem[i].block='p';
                        }
                        else if(i==2 || i==10 || i==18 || i==36 || i==54 || i==86
|| i==118)
                        {
                            elem[i].group=18;
                            elem[i].block='p';
                        }

                }
                for(i=57;i<=71;++i)
                {
                            elem[i].block='f';
                }
                for(i=89;i<=103;++i)
                {
                             elem[i].block='f';
                }
  }
   //Initializing period number.
   {
                for(i=1;i<=2;++i)
                {
                            elem[i].period=1;
                }
                for(i=3;i<=10;++i)
                {
                            elem[i].period=2;
                }
                for(i=11;i<=18;++i)
                {
                            elem[i].period=3;
                }
                for(i=19;i<=36;++i)
                {
                            elem[i].period=4;
                }
                for(i=37;i<=54;++i)
                {
                            elem[i].period=5;
                }
                for(i=55;i<=86;++i)
                {
```

```
                        elem[i].period=6;
                }
                for(i=87;i<=114;++i)
                {
                        elem[i].period=7;
                }
    }
    //User Interface coding.
    do
    {
    first:
    cout<<"\n\n\n\n\n\n\n\t\t\tWhat would you like to do?"<<"\n\n\n";
    Sleep(1000);
    cout<<"\t1. Search for an element\t";
    Sleep(1000);
    cout<<"2. View the Modern Periodic Table"<<"\n\n\n";
    Sleep(2000);
    cout<<"\t\t\tEnter your option: ";
    cin>>t;
    system("cls");
    if(t==1)
    {
    start:
    cout<<"\n\n\n\n\n\n\n\t\tHow would you like to search for your element?
"<<"\n\n\n";
    Sleep(1000);
    cout<<"\t1. Entering the atomic number\t";
    Sleep(1000);
    cout<<"2. Entering the element name"<<"\n\n\n";
    Sleep(2000);
    cout<<"\t\t\tEnter your option: ";
    cin>>n;
    system("cls");
    switch(n)
    {
            case 1: cout<<"\n\n\n\n\n\t\tEnter the atomic number of the element: ";
                    one:
                    cin>>a;
                    for(i=1;i<=114;++i)
                    {
                      if(a==i)
                      {
                      cout<<"\n\a";
                      cout<<"\t\t         Name            ";puts(elem[i].name);cout<<"\n";
                      cout<<"\t\t      Atomic Number      "<<elem[i].atomicno<<"\n";
                      cout<<"\t\t Electronic Configuration  ";puts(elem[i].ec);
                      cout<<"\t\t        Group            "<<elem[i].group<<"\n";
```

```
                cout<<"\t\t      Period          "<<elem[i].period<<"\n";
                cout<<"\t\t      Block           "<<elem[i].block<<"\n";
                cout<<"\t\t    Atomic Weight       "<<elem[i].atomicw<<"\n";
                if(elem[i].atomicr==0)
                {cout<<"\t\t    Atomic Radius       "<<"Not Available"<<"\n";}
                else
                {cout<<"\t\t    Atomic Radius       "<<elem[i].atomicr<<"
pm"<<"\n";}
                if(elem[i].en==0)
                {cout<<"\t\t  Electronegativity     "<<"Not Available"<<"\n";}
                else
                {cout<<"\t\t  Electronegativity     "<<elem[i].en<<" (Pauling
Scale)"<<"\n";}
                if(elem[i].meltp==0)
                {cout<<"\t\t    Melting Point       "<<"Not Available"<<"\n";}
                else
                {cout<<"\t\t    Melting Point       "<<elem[i].meltp<<" K"<<"\n";}
                if(elem[i].boilp==0)
                {cout<<"\t\t    Boiling Point       "<<"Not Available"<<"\n";}
                else
                {cout<<"\t\t    Boiling Point       "<<elem[i].boilp<<" K"<<"\n";}
                if(elem[i].density==0)
                {cout<<"\t\t      Density           "<<"Not Available"<<"\n";}
                else
                {cout<<"\t\t      Density           "<<elem[i].density<<"
g/cm3"<<"\n";}
                goto end;
                }
              }
              cout<<"\n\t\t\t  Please try again";
              Sleep(2000);
              system("cls");
              cout<<"\n\n\n\n\n\t\tEnter the atomic number of the element: ";
              goto one;
              break;
        case 2: cout<<"\t  Enter the element name given by the following format:
"<<"\n\n";
              cout<<"\t\t  \"Element Name\"- \"Element Symbol\""<<"\n\n";
              cout<<"\t\t  ";
              cin.getline(name,50);//Two cin statements to prevent keyboard
buffer.
              two:
              cin.getline(name,50);
              for(i=1;i<=114;++i)
              {
                if(strcmp(name,elem[i].name)==0)
                {
```

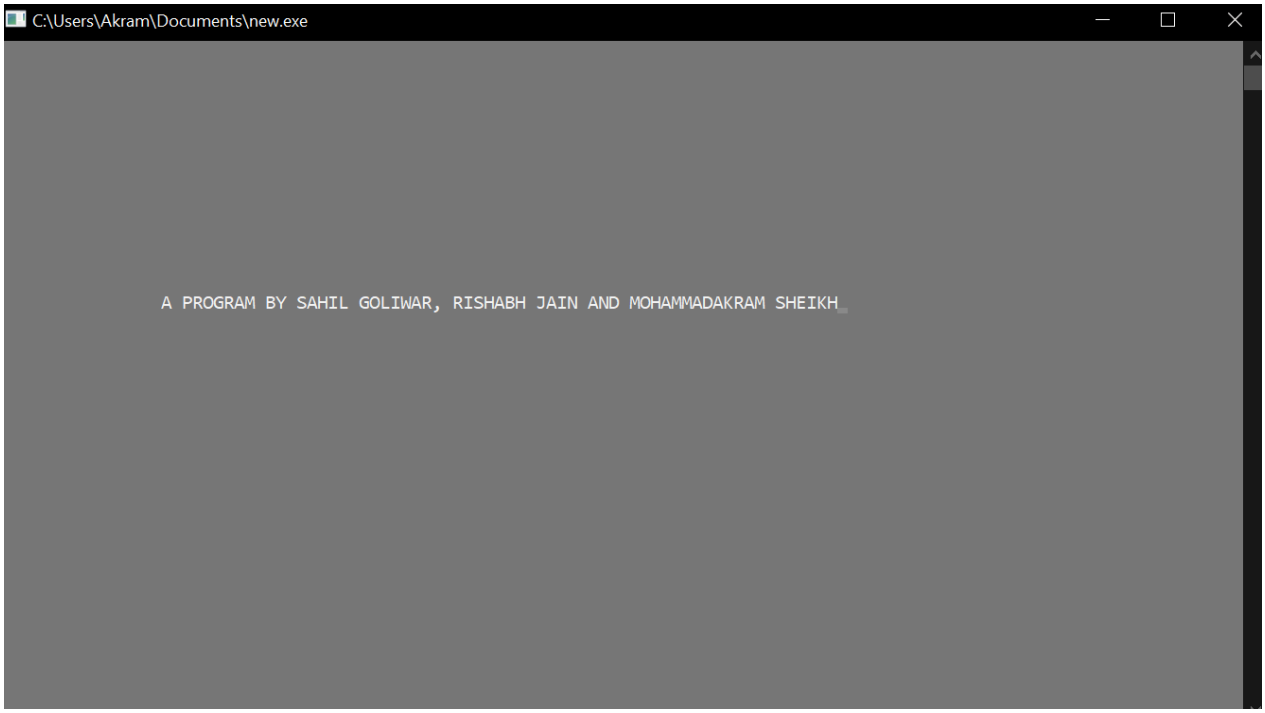
```
                cout<<"\n\a";
                cout<<"\t\t          Name          ";puts(elem[i].name);cout<<"\n";
                cout<<"\t\t      Atomic Number      "<<elem[i].atomicno<<"\n";
                cout<<"\t\t Electronic Configuration  ";puts(elem[i].ec);
                cout<<"\t\t         Group          "<<elem[i].group<<"\n";
                cout<<"\t\t         Period         "<<elem[i].period<<"\n";
                cout<<"\t\t         Block          "<<elem[i].block<<"\n";
                cout<<"\t\t      Atomic Weight      "<<elem[i].atomicw<<"\n";
                if(elem[i].atomicr==0)
                {cout<<"\t\t      Atomic Radius      "<<"Not Available"<<"\n";}
                else
                {cout<<"\t\t      Atomic Radius      "<<elem[i].atomicr<<"
pm"<<"\n";}
                if(elem[i].en==0)
                {cout<<"\t\t    Electronegativity    "<<"Not Available"<<"\n";}
                else
                {cout<<"\t\t    Electronegativity    "<<elem[i].en<<" (Pauling
Scale)"<<"\n";}
                if(elem[i].meltp==0)
                {cout<<"\t\t      Melting Point      "<<"Not Available"<<"\n";}
                else
                {cout<<"\t\t      Melting Point      "<<elem[i].meltp<<" K"<<"\n";}
                if(elem[i].boilp==0)
                {cout<<"\t\t      Boiling Point      "<<"Not Available"<<"\n";}
                else
                {cout<<"\t\t      Boiling Point      "<<elem[i].boilp<<" K"<<"\n";}
                if(elem[i].density==0)
                {cout<<"\t\t         Density         "<<"Not Available"<<"\n";}
                else
                {cout<<"\t\t         Density         "<<elem[i].density<<"
g/cm3"<<"\n";}
                goto end;
                }
              }
              cout<<"\t\t\t  Please try again";
              Sleep(2000);
              system("cls");
              cout<<"\t  Enter the element name given by the following format:
"<<"\n\n";
              cout<<"\t\t  \"Element Name\"- \"Element Symbol\""<<"\n\n";
              cout<<"\t\t  ";
              goto two;
              break;
      default: cout<<"\n\n\n\n\n\n\n\n\n\n\t\t\t\tWrong Option!";
              Sleep(2000);
              system("cls");
              goto start;
```

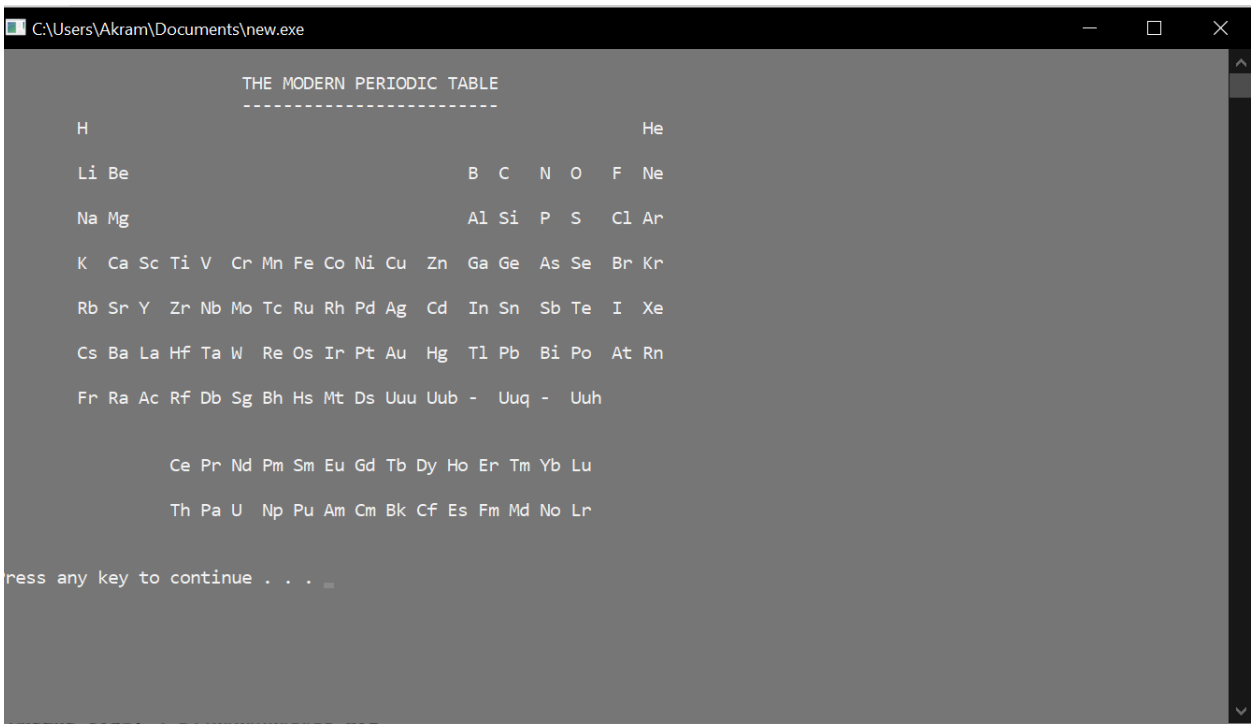
Modern Periodic Table

```
                break;
    }
    }


    else if(t==2)
    {
        cout<<"\n\t\t\tTHE MODERN PERIODIC TABLE\n";
        cout<<"\t\t\t-----------------------\n";
        cout<<"\tH                                                He"<<"\n\n";
        cout<<"\tLi Be                                B  C   N  O   F  Ne"<<"\n\n";
        cout<<"\tNa Mg                                 Al Si  P  S   Cl Ar"<<"\n\n";
        cout<<"\tK  Ca Sc Ti V  Cr Mn Fe Co Ni Cu  Zn  Ga Ge  As Se  Br Kr"<<"\n\n";
        cout<<"\tRb Sr Y  Zr Nb Mo Tc Ru Rh Pd Ag  Cd  In Sn  Sb Te  I  Xe"<<"\n\n";
        cout<<"\tCs Ba La Hf Ta W  Re Os Ir Pt Au  Hg  Tl Pb  Bi Po  At Rn"<<"\n\n";
        cout<<"\tFr Ra Ac Rf Db Sg Bh Hs Mt Ds Uuu Uub -  Uuq -  Uuh"<<"\n\n\n";
        cout<<"\t\t Ce Pr Nd Pm Sm Eu Gd Tb Dy Ho Er Tm Yb Lu"<<"\n\n";
        cout<<"\t\t Th Pa U  Np Pu Am Cm Bk Cf Es Fm Md No Lr"<<"\n\n\n";
        goto end;
    }
    else
    {
       cout<<"\n\n\n\n\n\n\n\n\n\n\t\t\t\tWrong Option!";
       Sleep(2000);
       system("cls");
       goto first;
    }

    end:
    cout<<"\n\n";
    system("PAUSE");
    system("cls");
    cout<<"\n\n\n\n\n\n\n\n\n\n\n\t\tWould you like to continue?(Y/N) ";
    cin>>c;
    if(c=='n' || c=='N')
    {
            exit(0);
    }
    system("cls");
    }while(c=='y' || c=='Y');

    getch();
    return 0;
}
```

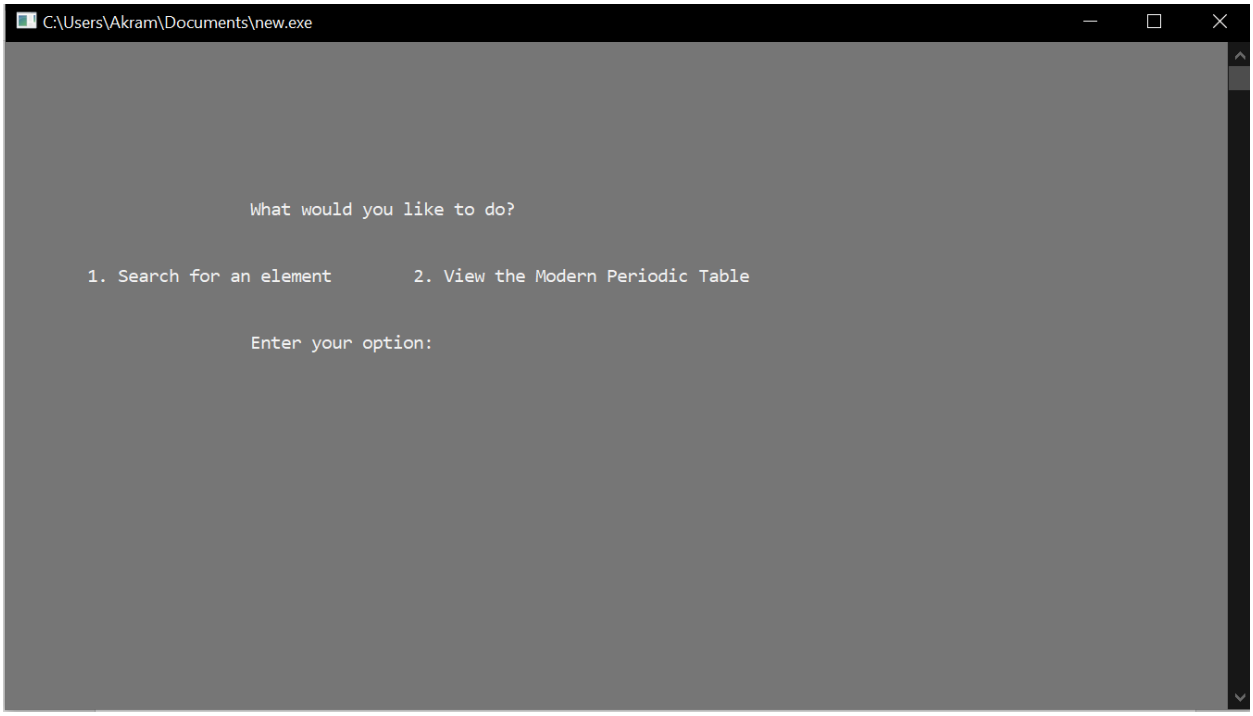
Modern Periodic Table

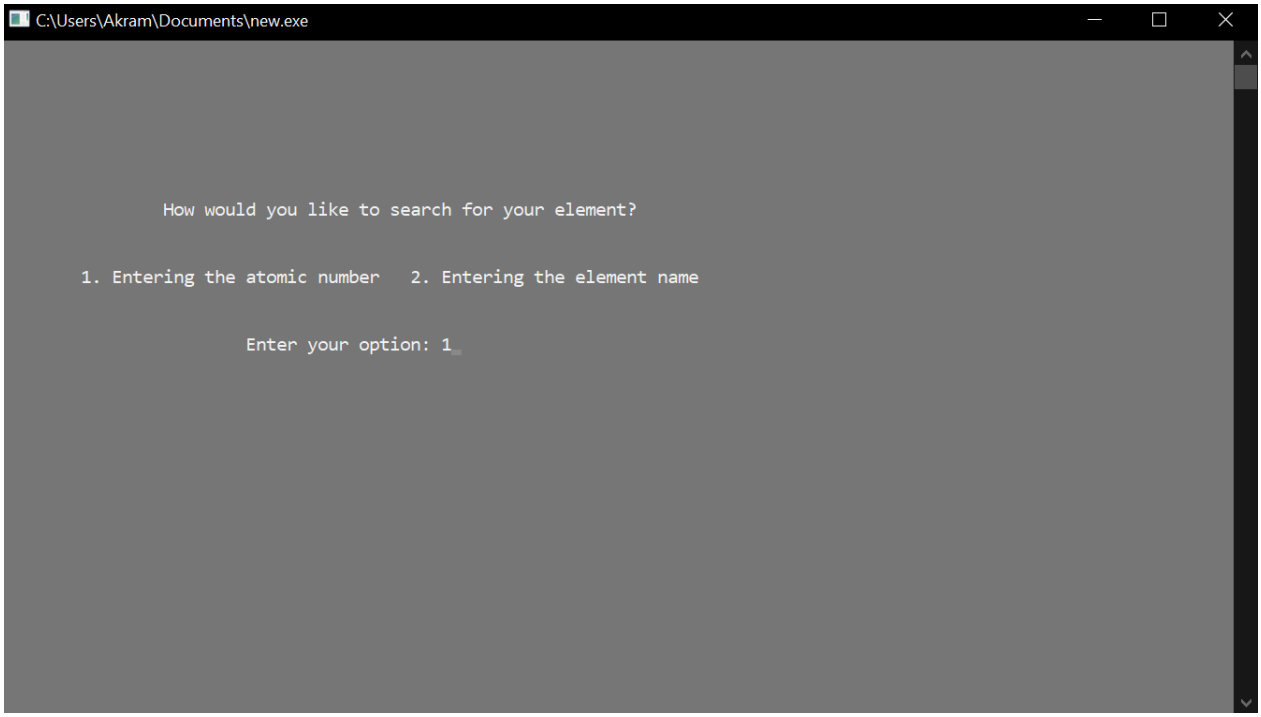
Screenshot:





Modern Periodic Table





Modern Periodic Table