Name: Aryan Patel

SIN: 301226774

using System;

using System.IO;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Serialization;

using System.Xml;

using System.Web.Script.Serialization;

namespace Week\_04\_lab\_12\_13

{

class Program

{

public static void complexDriver()

{

Complex c0 = new Complex(-2, 3);

Complex c1 = new Complex(-2, 3);

Complex c2 = new Complex(1, -2);

Console.WriteLine($"{c0}");

Console.WriteLine(c1);

Console.WriteLine(c2);

Console.WriteLine($"({c1}) + ({c2}) = {c1 + c2}");

Console.WriteLine($"({c1}) - ({c2}) = {c1 - c2}");

Console.WriteLine($"({c1}) \* ({c2}) = {c1 \* c2}");

Console.WriteLine($"- ({c1}) = {-c1}");

Complex c3 = c1 + c2;

Console.WriteLine($"{c3} in polar form is {c3.Modulus:f2}cis({ c3.Argument:f2})");

Console.WriteLine($"{c0} {(c0 == c1 ? "=" : "!=")} {c1}");

Console.WriteLine($"{c0} {(c0 == c2 ? "=" : "!=")} {c2}");

}

public static void atomsDriver()

{

List<Atom> atoms = Atom.GetAtoms();

// Display all the items in the collection elements

Console.WriteLine("Display all the items in the collection elements");

foreach (var atom in atoms)

{

Console.WriteLine(atom.ToString());

}

// Serialize the first item using xml format and save to a suitable file

Console.WriteLine("\n\nSerialize the first item using xml format and save to a suitable file");

TextWriter writer = new StreamWriter("atomXml2.xml");

XmlSerializer serializer = new XmlSerializer(typeof(Atom));

serializer.Serialize(writer, atoms[0]);

writer.Close();

// Read the above file and display the item

Console.WriteLine("\n\nRead the above file and display the item");

string xmldata = File.ReadAllText("atomXml2.xml");

Console.WriteLine(xmldata);

// Serialize the first item using json format and save to a suitable file

Console.WriteLine("\n\nSerialize the first item using json format and save to a suitable file");

TextWriter writerJ = new StreamWriter("atomJson.json");

writerJ.Write(new JavaScriptSerializer().Serialize(atoms[0]));

writerJ.Close();

// Read the above file and display the item

Console.WriteLine("\n\nRead the above file and display the item");

string jsonData = File.ReadAllText("atomJson.json");

Console.WriteLine(jsonData);

// Serialize the entire collection using json format and save to a suitable file

Console.WriteLine("\n\nSerialize the entire collection using json format and save to a suitable file");

Atom.WriteJson(atoms,"atomsListJson.json");

// Read the above file and display all the items

Console.WriteLine("\n\nRead the above file and display all the items");

string jsonListData = File.ReadAllText("atomsListJson.json");

Console.WriteLine(jsonListData);

}

static void Main(string[] args)

{

// complex class driver function

complexDriver();

// atom class driver function

atomsDriver();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Week\_04\_lab\_12\_13

{

class Complex

{

public int Real { get;}

public int Imaginary { get;}

public double Argument { get;}

public double Modulus { get;}

public static Complex Zero = new Complex();

public Complex (int real=0, int imaginary=0)

{

this.Real = real;

this.Imaginary = imaginary;

this.Modulus = Math.Sqrt((Math.Pow(this.Real, 2) + Math.Pow(this.Imaginary, 2)));

if (real == 0)

this.Argument = double.PositiveInfinity;

else this.Argument = Math.Atan(this.Imaginary / this.Real);

}

public override string ToString()

{

return $"{this.Real} + i({this.Imaginary})";

}

public static Complex operator +(Complex lhs, Complex rhs)

{

int real = lhs.Real + rhs.Real;

int imaginary = lhs.Imaginary + rhs.Imaginary;

return new Complex(real, imaginary);

}

public static Complex operator -(Complex lhs, Complex rhs)

{

int real = lhs.Real - rhs.Real;

int imaginary = lhs.Imaginary - rhs.Imaginary;

return new Complex(real, imaginary);

}

public static bool operator ==(Complex lhs, Complex rhs)

{

if (lhs.Real == rhs.Real && lhs.Imaginary == rhs.Imaginary)

return true;

else return false;

}

public static bool operator !=(Complex lhs, Complex rhs)

{

if (lhs.Real == rhs.Real && lhs.Imaginary == rhs.Imaginary)

return false;

else return true;

}

public static Complex operator \*(Complex lhs, Complex rhs)

{

int real = lhs.Real\*rhs.Real - lhs.Imaginary \* rhs.Imaginary;

int imaginary = lhs.Real \* rhs.Imaginary - rhs.Imaginary \* lhs.Real;

return new Complex(real, imaginary);

}

public static Complex operator -(Complex rhs)

{

return new Complex(-rhs.Real, -rhs.Imaginary);

}

}

}

using System;

using System.IO;

using System.Collections.Generic;

using System.Web.Script.Serialization;

using System.Text;

using Newtonsoft.Json.Linq;

namespace Week\_04\_lab\_12\_13

{

public class Atom

{

public string Name { get; set; }

public string Symbol { get; set; }

public int Proton { get; set; }

public int Neutron { get; set; }

public double Weight { get; set; }

public Atom()

{ }

public Atom(string name, int proton, int neutron, double weight, string symbol )

{

this.Name = name;

this.Proton = proton;

this.Neutron = neutron;

this.Weight = weight;

this.Symbol = symbol;

}

public static Atom Parse(string line)

{

string[] ObjectData = line.Split(' ');

if (ObjectData.Length != 5)

throw new ArgumentException("string Array should have 5 elements");

else

{

return new Atom(ObjectData[0], Int32.Parse(ObjectData[1]), Int32.Parse(ObjectData[2]), Double.Parse(ObjectData[3]), ObjectData[4]);

}

}

public static void WriteJson(List<Atom> atoms, string filename)

{

TextWriter writer = new StreamWriter(filename);

writer.Write(new JavaScriptSerializer().Serialize(atoms));

writer.Close();

}

public static void ReadJson(string filename)

{

TextReader reader = new StreamReader(filename);

string jsonString = reader.ReadToEnd();

JObject obj = JObject.Parse(jsonString);

var jsonArray = JArray.Parse(obj["atoms"].ToString());

}

public static List<Atom> GetAtoms()

{

List<Atom> elements = new List<Atom>();

elements.Add(Atom.Parse("Hydrogen 1 0 1.0079 H"));

elements.Add(Atom.Parse("Helium 2 2 4.0026 He")); ;

elements.Add(Atom.Parse("Lithium 3 4 6.941 Li")); ;

elements.Add(Atom.Parse("Beryllium 4 5 9.0122 Be"));

elements.Add(Atom.Parse("Boron 5 6 10.811 B"));

elements.Add(Atom.Parse("Carbon 6 6 12.0107 C"));

elements.Add(Atom.Parse("Nitrogen 7 7 14.0067 N"));

elements.Add(Atom.Parse("Oxygen 8 8 15.9994 O"));

elements.Add(Atom.Parse("Fluorine 9 10 18.9984 F"));

elements.Add(Atom.Parse("Neon 10 10 20.1797 Ne"));

elements.Add(Atom.Parse("Sodium 11 12 22.9897 Na"));

elements.Add(Atom.Parse("Magnesium 12 12 24.305 Mg"));

elements.Add(Atom.Parse("Aluminum 13 14 26.9815 Al"));

elements.Add(Atom.Parse("Silicon 14 14 28.0855 Si"));

elements.Add(Atom.Parse("Phosphorus 15 16 30.9738 P"));

elements.Add(Atom.Parse("Sulfur 16 16 32.065 S"));

elements.Add(Atom.Parse("Chlorine 17 18 35.453 Cl"));

elements.Add(Atom.Parse("Potassium 19 20 39.0983 K"));

elements.Add(Atom.Parse("Argon 18 22 39.948 Ar"));

elements.Add(Atom.Parse("Calcium 20 20 40.078 Ca"));

elements.Add(Atom.Parse("Scandium 21 24 44.9559 Sc"));

elements.Add(Atom.Parse("Titanium 22 26 47.867 Ti"));

elements.Add(Atom.Parse("Vanadium 23 28 50.9415 V"));

elements.Add(Atom.Parse("Chromium 24 28 51.9961 Cr"));

elements.Add(Atom.Parse("Manganese 25 30 54.938 Mn"));

elements.Add(Atom.Parse("Iron 26 30 55.845 Fe"));

elements.Add(Atom.Parse("Nickel 28 31 58.6934 Ni"));

elements.Add(Atom.Parse("Cobalt 27 32 58.9332 Co"));

elements.Add(Atom.Parse("Copper 29 35 63.546 Cu"));

elements.Add(Atom.Parse("Zinc 30 35 65.39 Zn"));

elements.Add(Atom.Parse("Gallium 31 39 69.723 Ga"));

elements.Add(Atom.Parse("Germanium 32 41 72.64 Ge"));

elements.Add(Atom.Parse("Arsenic 33 42 74.9216 As"));

elements.Add(Atom.Parse("Selenium 34 45 78.96 Se"));

elements.Add(Atom.Parse("Bromine 35 45 79.904 Br"));

elements.Add(Atom.Parse("Krypton 36 48 83.8 Kr"));

elements.Add(Atom.Parse("Rubidium 37 48 85.4678 Rb"));

elements.Add(Atom.Parse("Strontium 38 50 87.62 Sr"));

elements.Add(Atom.Parse("Yttrium 39 50 88.9059 Y"));

elements.Add(Atom.Parse("Zirconium 40 51 91.224 Zr"));

elements.Add(Atom.Parse("Niobium 41 52 92.9064 Nb"));

elements.Add(Atom.Parse("Molybdenum 42 54 95.94 Mo"));

elements.Add(Atom.Parse("Technetium 43 55 98 Tc"));

elements.Add(Atom.Parse("Ruthenium 44 57 101.07 Ru"));

elements.Add(Atom.Parse("Rhodium 45 58 102.9055 Rh"));

elements.Add(Atom.Parse("Palladium 46 60 106.42 Pd"));

elements.Add(Atom.Parse("Silver 47 61 107.8682 Ag"));

elements.Add(Atom.Parse("Cadmium 48 64 112.411 Cd"));

elements.Add(Atom.Parse("Indium 49 66 114.818 In"));

elements.Add(Atom.Parse("Tin 50 69 118.71 Sn"));

elements.Add(Atom.Parse("Antimony 51 71 121.76 Sb"));

elements.Add(Atom.Parse("Iodine 53 74 126.9045 I"));

elements.Add(Atom.Parse("Tellurium 52 76 127.6 Te"));

elements.Add(Atom.Parse("Xenon 54 77 131.293 Xe"));

elements.Add(Atom.Parse("Cesium 55 78 132.9055 Cs"));

elements.Add(Atom.Parse("Barium 56 81 137.327 Ba"));

elements.Add(Atom.Parse("Lanthanum 57 82 138.9055 La"));

elements.Add(Atom.Parse("Cerium 58 82 140.116 Ce"));

elements.Add(Atom.Parse("Praseodymium 59 82 140.9077 Pr"));

elements.Add(Atom.Parse("Neodymium 60 84 144.24 Nd"));

elements.Add(Atom.Parse("Promethium 61 84 145 Pm"));

elements.Add(Atom.Parse("Samarium 62 88 150.36 Sm"));

elements.Add(Atom.Parse("Europium 63 89 151.964 Eu"));

elements.Add(Atom.Parse("Gadolinium 64 93 157.25 Gd"));

elements.Add(Atom.Parse("Terbium 65 94 158.9253 Tb"));

elements.Add(Atom.Parse("Dysprosium 66 97 162.5 Dy"));

elements.Add(Atom.Parse("Holmium 67 98 164.9303 Ho"));

elements.Add(Atom.Parse("Erbium 68 99 167.259 Er"));

elements.Add(Atom.Parse("Thulium 69 100 168.9342 Tm"));

elements.Add(Atom.Parse("Ytterbium 70 103 173.04 Yb"));

elements.Add(Atom.Parse("Lutetium 71 104 174.967 Lu"));

elements.Add(Atom.Parse("Hafnium 72 106 178.49 Hf"));

elements.Add(Atom.Parse("Tantalum 73 108 180.9479 Ta"));

elements.Add(Atom.Parse("Tungsten 74 110 183.84 W"));

elements.Add(Atom.Parse("Rhenium 75 111 186.207 Re"));

elements.Add(Atom.Parse("Osmium 76 114 190.23 Os"));

elements.Add(Atom.Parse("Iridium 77 115 192.217 Ir"));

elements.Add(Atom.Parse("Platinum 78 117 195.078 Pt"));

elements.Add(Atom.Parse("Gold 79 118 196.9665 Au"));

elements.Add(Atom.Parse("Mercury 80 121 200.59 Hg"));

elements.Add(Atom.Parse("Thallium 81 123 204.3833 Tl"));

elements.Add(Atom.Parse("Lead 82 125 207.2 Pb"));

elements.Add(Atom.Parse("Bismuth 83 126 208.9804 Bi"));

elements.Add(Atom.Parse("Polonium 84 125 209 Po"));

elements.Add(Atom.Parse("Astatine 85 125 210 At"));

elements.Add(Atom.Parse("Radon 86 136 222 Rn"));

elements.Add(Atom.Parse("Francium 87 136 223 Fr"));

elements.Add(Atom.Parse("Radium 88 138 226 Ra"));

elements.Add(Atom.Parse("Actinium 89 138 227 Ac"));

elements.Add(Atom.Parse("Protactinium 91 140 231.0359 Pa"));

elements.Add(Atom.Parse("Thorium 90 142 232.0381 Th"));

elements.Add(Atom.Parse("Neptunium 93 144 237 Np"));

elements.Add(Atom.Parse("Uranium 92 146 238.0289 U"));

elements.Add(Atom.Parse("Americium 95 148 243 Am"));

elements.Add(Atom.Parse("Plutonium 94 150 244 Pu"));

elements.Add(Atom.Parse("Curium 96 151 247 Cm"));

elements.Add(Atom.Parse("Berkelium 97 150 247 Bk"));

elements.Add(Atom.Parse("Californium 98 153 251 Cf"));

elements.Add(Atom.Parse("Einsteinium 99 153 252 Es"));

elements.Add(Atom.Parse("Fermium 100 157 257 Fm"));

elements.Add(Atom.Parse("Mendelevium 101 157 258 Md"));

elements.Add(Atom.Parse("Nobelium 102 157 259 No"));

elements.Add(Atom.Parse("Rutherfordium 104 157 261 Rf"));

elements.Add(Atom.Parse("Lawrencium 103 159 262 Lr"));

elements.Add(Atom.Parse("Dubnium 105 157 262 Db"));

elements.Add(Atom.Parse("Bohrium 107 157 264 Bh"));

elements.Add(Atom.Parse("Seaborgium 106 160 266 Sg"));

elements.Add(Atom.Parse("Meitnerium 109 159 268 Mt"));

elements.Add(Atom.Parse("Roentgenium 111 161 272 Rg"));

elements.Add(Atom.Parse("Hassium 108 169 277 Hs"));

return elements;

}

public override string ToString()

{

return $"Name: {this.Name}\nproton: {this.Proton}\nneutron: {this.Neutron}\nweight: {this.Weight}\nsymbole: {this.Symbol}\n\n";

}

}

}