AMERICAN UNIVERSITY OF ARMENIA
College of Science and Engineering
CS 140 Mechanics
PROJECT - Springs

PROGRESS EVALUATIONS: Thursday April 13 2023, Tuesday April 18 2023
FINAL SUBMISSION DEADLINE: Tuesday, April 25 2023, not later than 23:59

CONTACTS: Suren Khachatryan, skhachat@aua.am
Tsovinar Karapetyan, tsovinar.karapetyan.0510@gmail.com

Project Objective

The project aims at design and simulation of a mechanical converter that converts binary representation of a number into its value in the decimal numeral system. The device gets as its input a sequence of bits and generates a spring oscillations the frequency of which corresponds to the magnitude of the recorded input.

Task 1 (reading: **D. Kleppner, R. Kolenkow. "Introduction to_Mechanics"**, 2nd ed., 2014 – see Kleppner_Kolenkow_Introduction_to_Mechanics_2ed_2014.pdf in References.zip archive, **Chapter 3**. *Forces and Equations of Motion*, **section 3.7** *Hook's Law and Simple Harmonic Motion*, pages 102-107)

Write a class Spring that implements the concept of a 1D massless spring and, hence, encapsulates its stiffness **double** *k* with the default value equal 1. Add the following methods:

1. The default constructor and an overloaded constructor that specifies the stiffness;
2. The public getter and a private setter;
3. Overloaded public *move()* methods that return an array of coordinates of an oscillating mass:
   - **double**[] *move*(**double** *t*, **double** *dt*, **double** *x0*, **double** *v0*) – a body of unit mass oscillates during a period **double** *t* starting from $t = 0$ with initial conditions $x(0) = x0$ and $v(0) = v0$. The coordinate is computed per each **double** *dt* time step;
   - **double**[] *move*(**double** *t*, **double** *dt*, **double** *x0*) – a body of unit mass oscillates during a period **double** *t* starting from $t = 0$ with initial conditions $x(0) = x0$ and $v(0) = 0$. The coordinate is computed per each **double** *dt* time step;
   - **double**[] *move*(**double** *t0*, **double** *t1*, **double** *dt*, **double** *x0*, **double** *v0*) – a body of unit mass oscillates from $t = t0$ till $t = t1$ with initial conditions $x(t0) = x0$ and $v(t0) = v0$. The coordinate is computed per each **double** *dt* time step;
   - **double**[] *move*(**double** *t0*, **double** *t1*, **double** *dt*, **double** *x0*, **double** *v0*, **double** *m*) – a body of a specified mass **double** *m* oscillates from $t = t0$ till $t = t1$ with initial conditions $x(t0) = x0$ and $v(t0) = v0$. The coordinate is computed per each **double** *dt* time step.

Task 2 (reading: **D. Kleppner, R. Kolenkow. "Introduction to_Mechanics"**, 2nd ed., 2014 – see Kleppner_Kolenkow_Introduction_to_Mechanics_2ed_2014.pdf in References.zip archive, **Chapter 3**. *Forces and Equations of Motion*, **section 3.7** *Hook's Law and Simple Harmonic Motion*, pages 102-107)

1. Continue with the class Spring and add the following public methods:

- Spring *inSeries*(Spring *that*) – takes by reference a Spring *that* argument, connects it with **this** Spring in series and returns a new Spring object that represents the equivalent spring;
- Spring *inParallel*(Spring *that*) – takes by reference a Spring *that* argument, connects it with **this** Spring in parallel and returns a new Spring object that represents the equivalent spring;

2. Write a class SpringArray and implement the following public static methods:
   - Spring *equivalentSpring*(String *springExpr*) – takes a String expression that represents connections of springs of unit stiffness and returns the equivalent spring. The Spring *springExpr* is a valid expression of balanced braces {} and brackets []. Empty braces or brackets without nested braces and brackets represent a single spring of unit stiffness. Brackets with nested braces and brackets represent springs connected in parallel. Braces with nested braces and brackets represent springs connected in series.
   - Spring *equivalentSpring*(String *springExpr*, Spring[] *springs*) – takes a String expression that represents connections of springs specified by a Spring array Spring[] *springs* and returns the equivalent spring.

Task 3 (reading: **H. Gould, J. Tobochnik, W. Christian. "An Introduction to Computer Simulation Methods: Applications to Physical System"**, 3rd ed., 2011 – see Gould_Tobochnik_Christian_Intro_to_Comp_Simulation_Methods_3rd_ed_2011.pdf in References.zip archive, **Chapter 9**. *Normal Modes and Waves*, **section 9.3** *Fourier Series*, pages 319-329, **section 9.5** *Fourier Integrals*, pages 331-332, **Appendix 9A** *Complex Fourier Series*, **Appendix 9B** *Fast Fourier Transform*, pages 353-357)

Write a class FT that implements the concept of Fourier transform / series. It transforms an array of coordinate values at different time moments into an array of the amplitudes of harmonic oscillations. Declare member variables and implement methods as needed.

Task 4

Write an abstract **class** *Converter* that converts binary representation of a number into its value in the decimal numeral system.

- Declare an abstract method for taking as its argument a sequence of bits and returning the corresponding system of springs.
- Add a concrete method that connects to the system of springs a body of unit mass and computes its oscillations.
- Add a concrete method that calculates the frequency amplitudes of the oscillations using the implemented Fourier Transform.
- Declare an abstract method for evaluation of the decimal value of the original binary sequence using the computed frequency amplitudes.

Task 5

Derive or extend a concrete **class** *Converter8Bit* that converts binary representation of an integer number into its value in the decimal numeral system. Consider sequences of 8 bits and design systems of unit springs that implement each of them.

- Override the inherited abstract methods.
- Test the implemented *Converter8Bit* class for several cases in the **main**() function.

Task 6

Derive or extend a concrete **class** *ConverterInt* that converts binary representation of an integer number into its value in the decimal numeral system. Consider bit sequences of arbitrary length and minimize the number of the used unit springs.

- Override the inherited abstract methods.
- Test the implemented *ConverterInt* class for several cases in the **main**() function.

Task 7

Derive or extend a concrete class *ConverterFloat* that converts binary representation of a floating point number into its value in the decimal numeral system. The number is represented by two bit sequences of arbitrary lengths. The first sequence represents the integer part and the second one – the fractional part.

- Override the inherited abstract methods.
- Test the implemented *ConverterFloat* class for several cases in the **main**() function.

Submission Format and Project Requirements

1. Before starting the project create a repository to keep there all project deliverables. Share it with Tsovinar Karapetyan (tsovinar.karapetyan.0510@gmail.com) and. Suren Khachatryan (skhachat@aua.am).
2. The project may be implemented in any OOP environment.
3. The project deadline is Tuesday April 25, not later than 23:59. No special submission is needed – the project will be evaluated based on the repository status at the moment of the deadline.
4. Two progress evaluations will be conducted before the final deadline – on Saturday April 30 and Friday May 06. Again, no special submissions are needed – the progress will be evaluated based on the ongoing repository status and a short free-format status report. The reports list all project-related activities, including understanding, study, reading, design, development, testing, analysis, etc.
5. This is an individual project. Therefore, the individual contributions must be explicitly stated in the project docs.
6. You are welcome to use / reuse / consult external (re)sources, including open-source code, electronic and hard-copy texts, videos, group discussions, instructor / TA assistance, etc. All such sources must be explicitly acknowledged / referenced. Any unreferenced use of external sources will violate the rules of the academic integrity.