Object-Oriented Programming in C++ (SS 2019)
Exercise 6: June 20, 2019

Wolfgang Schreiner
Research Institute for Symbolic Computation (RISC)
Wolfgang.Schreiner@risc.jku.at

April 1, 2019

The exercise is to be submitted by the denoted deadline via the submission interface of the Moodle course as a single file in zip (`.zip`) or tarred gzip (`.tgz`) format which contains the following files:

- A PDF file `Exercise`*`Number`*`-`*`MatNr`*`.pdf` (where *Number* is the number of the exercise and *MatNr* is your "Matrikelnummer") which consists of the following parts:
  1. A decent cover page with the title of the course, the number of the exercise, and the author of the solution (identified by name, Matrikelnummer and email address).
  2. For every source file, a listing in a *fixed width font*, e.g. `Courier`, (such that indentations are appropriately preserved) and an appropriate *font size* such that source code lines to not break.
  3. A description of all tests performed (copies of program inputs and program outputs) explicitly highlighting, if some test produces an unexpected result.
  4. Any additional explanation you would like to give. In particular, if your solution has unwanted problems or bugs, please document these explicitly (you will get more credit for such solutions).
- Each source file of your solution (no object files or executables).

Please obey the coding style recommendations posted on the course site.

Exercise 6: Polygons by Containers

The goal of this exercise is to generalize the classes developed in Exercise 3 where coordinates where represented by `double` values to class templates that take a type parameter *C* that represents the type of the coordinates:

```
template<typename C> class Point ... ;
template<typename C> class Polygon ... ;
template<typename C> class PenPolygon ... ;
template<typename C> class SpiralPolygon ... ;
template<typename C> class Picture ... ;
```

Here *C* is assumed to be a numeric type with the usual arithmetic operations.

The internal representation of these objects shall be based on containers of the standard library; in particular, use types

```
list<Point<C> >
unordered_set<Polygon<C>*>
```

for the sequence of points in `Polygon`<*C*> and for the set of polygon pointers in `Picture`<*C*>, respectively (`unordered_set` is a variant of `set` that does not demand any ordering from its base type). Both kinds of collections are to be traversed by the corresponding iterators.

Test these classes as in Exercise 3 by creating a picture of spirals; instantiate type parameter *C* with argument `double`.