Computer Systems (SS 2011)
Exercise 5: June 6, 2011

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

May 17, 2011

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 do 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 5: Polygons by Sequence Containers

Take the template classes `Math`, `Point`, and `Lines` developed in Exercise 4 and implement the following abstract class for a polygon whose coordinates have some type C (which we assume to support the same operations as in Exercise 4).

```
template<typename C> class Polygon
{
public:
  virtual ~Polygon() { }

  // add point x,y to polygon
  virtual void add(C x, C y) = 0;

  // get number of points
  virtual int number() = 0;

  // get point number i, 0 <= i < number()
  virtual Point<C> point(int i) = 0;

  // the framework functions
  void random(int n, int x, int y, int w, int h, int seed = 0);
  bool read(const char* filename);
  void draw(unsigned int color1 = 0, unsigned int color 2 = 0);
};
```

The class represents a framework for generating a random polynomial, reading a polynomial from a file, and drawing a polynomial (including the intersection points). The class does not contain a concrete representation of the polygon but calls the abstract functions `add()`, `number()`, and `point()` to implement the non-abstract functions `random()`, `read()`, and `draw()`.

Derive from `Polygon` a non-abstract template class

```
template<typename C> class VectorPolygon: public Polygon<C> {...};
```

that provides concrete definitions for the inherited abstract functions; the class represents the polygon with the help of the standard library by an object of type `vector< Point<C> >`; the implementation shall as far as possible make use of the operations that are already available on this type.

Likewise, derive from `Polygon` a non-abstract template class

```
template<typename C> class DequePolygon: public Polygon<C> {...};
```

that represents the polygon as an object of type `deque< <Point<C> >`.

Please note that by the use of the standard library classes, it is not necessary to explicitly allocate heap memory with the operator `new`; thus there is also no need to re-define the default copy constructors, copy assignment operators, and destructors of these classes.

Finally write a class

```
template<typename C> PolygonSequence: public list< Polygon<C>* >
{
public:
  void draw(unsigned int color1 = 0, unsigned int color 2 = 0);
}
```

that represents a sequence of polygons and whose function `draw()` draws all polygons in the sequence. The class inherits its representation from `list< Polygon<C>* >` (i.e. pointers to polygon objects are stored) and does not contain any additional data.

Test the classes by creating multiple polygons of types `VectorPolygon<double>` and `DequePolygon<double>`, storing all polygons in a sequence `seq`, and drawing the sequence by a call of `seq.draw()`.