General remarks. Unless otherwise specified prepare solution in C++. Parts of the exercise 7 are considered to be sufficient individual exercises. Avoid duplication. Recommended tools to check code are GNU C++ compiler (`g++`), GHC (`ghc`/`ghci`), SML/NJ (`sml`).

Exercises:

1. Implement in C++ 'function' that takes polymorphic function as an argument[1] Is it possible to pass function template into such construction? What is the main difference with Haskell? Why `std::function` objects cannot be used?

2. Let α be a function with $\overline{0,n} \subseteq \delta_\alpha$, $\rho_\alpha \subseteq S$. Let F be a higher-order function that processes $\alpha_{\overline{0,n}}$ iterating function $f : T \times S \to T$ as follows:

$$\begin{cases} F([\,]) & = t_0 \\ F(\alpha_{\overline{0,k}} : a) & = f(F(\alpha_{\overline{0,k}}), a) \end{cases},$$

   where $[\,]$ is an empty sequence, ':' is an operation of adding element to the end of the sequence, $t_0 \in T$ is some constant. Propose an approach to implement F without extra memory allocation. Apply it to implement higher-order functions `min`, `max`, `sum` using STL algorithms.

3. Implement compile-time function `ent_of_double` for checking if the type `T` (considered) as a model of the concept `ent` has associated type `cotype` equal to `double` (function template without arguments – `bool ent_of_double<T>()`).

4. Implement structure with static operations: `mult(x)`, `div(x)`, `mod(x)`, that calculate respectively multiplication, quotient and reminder of division of x by constant n. Provide operation optimization for certain values of n.

5. Consider increasing n-tuples of numbers with upper bound m.[2] These tuples are totally ordered (lexicographically), so there are ranges of tuples. Let F be a function $F : \mathrm{int}^n \to S$. Without using "extra" space allow usage of STL algorithms like `min_element`, `accumulate` for ranges $\{Fx \mid x \in [\alpha, \beta)\}$, where α, β are tuples.

6. Consider n-tuples of numbers with upper bounds $\{m_i\}_{i=\overline{1,n}}$. These tuples are totally ordered (lexicographically), so there are ranges of tuples. Let F be a function $F : \mathrm{int}^n \to S$. Without using "extra" space allow usage of STL algorithms like `min_element`, `accumulate` for ranges $\{Fx \mid x \in [\alpha, \beta)\}$, where α, β are tuples.

7. Solve the problem of optimal order of matrices multiplication

   (a) using STL algorithm `std::min_element`.

   (b) using STL algorithm `std::accumulate`.

   Avoid extra space allocation, provide parenthesization

8. Follow example in Haskell[3] to implement fixed-point operator in Standard ML. Explain why it is not working. Changing underlying lambda term and its implementation obtain working version of fixed point operator.

9. Using `std::function` and anonymous functions implement Haskell example of `fixpointY` in C++ (make it compilable). Is it working?

10. In Haskell use `fixpointY` to construct another fixed-point operator as a fixed-point itself.

[1] example from `poly.pdf` on slides 15, 33

[2] i.e. monotone functions $\alpha : \overline{1,n} \to \overline{1,m}$

[3] in `simple.pdf` slides 42-43