Programowanie funkcyjne – laboratorium 7
25.11.2009 r.

1. Dane są następujące definicje.

```
type order = Less | Equal | Greater

module type SET =
  sig
    type elem
    type set
    val empty : set
    val insert : set -> elem -> set
    val contains : 'a set -> 'a -> bool
    val union : set -> set -> set
  end

module type ORD_TYPE =
  sig
    type t
    val cmp : t -> t -> order
  end

module type SET_FUNCTOR = functor (Ord : ORD_TYPE) -> SET
```

   Zaimplementuj zbiór na kilka sposobów:

   (a) zbiór jako lista uporządkowana,

   (b) zbiór jako funkcja charakterystyczna,

   (c) zbiór jako drzewo BST.

2. Dana jest (nowa) sygnatura słownika.

```
module type DICT =
  sig
    type key
    type value
    type dict

    exception Undefined of key

    val empty : dict
    val add : dict -> key -> value -> dict
    val find : dict -> key -> value
  end
```

   Zaimplementuj słownik w dowolny sposób jako funktor o dwóch parametrach. Na przykład:

```
module type VALUE =
  sig
    type t
  end
```

```
module ListDict (Key : ORD_TYPE) (Elem : VALUE) : DICT =
  struct
    type key = Key.t
    type value = Elem.t
    type dict = (key * value) list
    ...
  end
```