# A. Funkcje do obsługi kolekcji

W tej części przedstawiony jest spis funkcji znajdujących się w modułach `List`, `Seq`, `Map`, `Set`.

## A.1 Listy i sekwencje - moduły: List i Seq

Ponieważ listy i sekwencje zawierają bardzo podobny zestaw funkcji zostały one przedstawione razem.

1. **`List.allPairs lista1 lista2`**
   **`Seq.allPairs źródło1 zdrodlo2`**
   *sygnatura*: `'a list->'b list->('a*'b) list`
   `seq<'a>->seq<'b>->seq<('a*'b)>`
   *opis*: Zwraca nową kolekcję zawierającą wszystkie możliwe pary elementów kolekcji wejściowych
2. **`List.append lista1 lista2`**
   **`Seq.append źródło1 źródło2`**
   *sygnatura*: `'a list->'a list->'a list`
   `seq<'a>->seq<'a>->seq<'a>`
   *opis*: Zwraca nową kolekcję zawierającą elementy z pierwszej po których występują elementy z drugiej
3. **`List.average lista`**
   **`Seq.average źródło`**
   *sygnatura*: `^a list->^a`
   `seq<^a>->seq<^a>`
   *opis*: Oblicza średnią elementów kolekcji

4. `List.averageBy projektor lista`
   `Seq.averageBy projektor źródło`
   *sygnatura*: `('a->^b)->'a list->^b`
   `('a->^b)->seq<'a>->^b`
   *opis*: Oblicza średnią wartości wygenerowanych poprzez zastosowanie funkcji do każdego elementu kolekcji wejściowej
5. `Seq.cache źródło`
   *sygnatura*: `seq<'a>->seq<'a>`
   *opis*: Zwraca sekwencję, która odpowiada zbuforowanej wersji sekwencji wejściowej. Dzięki temu sekwencja może być przeglądana wiele razy bez konieczności ponowenego uruchamiania sekwekcji wejściowej
6. `Seq.cast źródło`
   *sygnatura*: `IEnumerable->seq<'a>`
   *opis*: Zamienia sekwencję obiektów na sekwencję typowaną. Wymaga wskazania typu elementów wyjściowych
7. `List.choose wyborca lista`
   `Seq.choose wyborca źródło`
   *sygnatura*: `'a->'b option->'a list->'b list`
   `'a->'b option->seq<'a>->seq<'b>`
   *opis*: Aplikuje funkcję do każdego elementu kolekcji. Następnie zwraca nową kolekcję zawierającą te elementy kolekcji wejściowej, dla której funkcja zwraca `Some(x)`
8. `List.chunkBySize rozmiar lista`
   `Seq.chunkBySize rozmiar źródło`
   *sygnatura*: `int->seq<'a> ->seq<'a[]>`
   *opis*: Dzieli kolekcję na fragmenty o maksymalnym rozmiarze rozmiar
9. `List.collect mapa lista`
   `Seq.collect mapa źródło`
   *sygnatura*: `'a->'b list->'a list->'b list`
   `'a->seq<'b>->seq<'a>->seq<'b>`
   *opis*: Aplikuje funkcję do każdego elementu kolekcji. Łączy wszystkie wyniki w jedną kolekcję i zwraca ją
10. `List.compareWith komparator lista1 lista2`
    `Seq.compareWith komparator źródło1 źródło2`
    *sygnatura*: `'a->'a->bool->'a list->'a list->int`
    `'a->'a->bool->seq<'a>->seq<'a>->int`
    *opis*: Porównuje dwie kolekcje element po elemencie. Zwraca pierwszy niezerowy wynik porównania. Jeżeli element w pierwszej kolekcji jest większy to zwraca wartość dodatnią. Jeżeli element w drugiej kolekcji jest większy to zwraca wartość ujemną. Jeżeli elementy są takie same w obu kolekcjach, to zwraca 1 - gdy pierwsza kolekcja jest dłuższa, 0 - gdy obie kolekcje mają tę samą długość, -1 - gdy druga kolekcja jest dłuższa.

11. `List.concat listy`
    `Seq.concat źródła`
    *sygnatura*: `seq<'a list>->'a list`
    `seq<'collection>->seq<'a>`
    *opis*: Zwraca nową kolekcję, która zawiera elementy z każdej kolekcji wejściowej w kolejności

12. `List.contains wartość lista`
    `Seq.contains wartość kolekcja`
    *sygnatura*: `'a->'a list->bool`
    `'a->seq<'a>->bool`
    *opis*: Sprawdza czy podany element znajduje się w kolekcji

13. `List.countBy projektor lista`
    `List.countBy projektor źródło`
    *sygnatura*: `'a->'b->'a list->('b*int) list`
    `'a->'b->seq<'a>->seq<('b*int)>`
    *opis*: Aplikuje funkcję generującą klucze, do każdego elementu kolekcji. Następnie zwraca kolekcję zawierającą unikalną wartość klucza oraz liczbę jego wystąpień w oryginalnej kolekcji

14. `Seq.delay generator`
    *sygnatura*: `unit->seq<'a>->seq<'a>`
    *opis*: Zwraca kolekcję, która jest zbudowana zgodnie z odroczoną specyfikacją sekwencji

15. `List.distinct lista`
    `Seq.distinct źródło`
    *sygnatura*: `seq<'a>->seq<'a>`
    *opis*: Zwraca kolekcję zawierającą tylko elementy unikalne zgodnie z ogólną metodą określania równości elementów

16. `List.distinctBy projektor lista`
    `Seq.distinctBy projektor kolekcja`
    *sygnatura*: `'a->'klucz->seq<'a>->seq<'a>`
    *opis*: Zwraca kolekcję zawierającą tylko elementy unikalne zgodnie z ogólną metodą określania równości elementów. Porównywane są klucze wygenerowane przez dostarczoną funkcję

17. `List.empty`
    `Seq.empty`
    *sygnatura*: `'a list`
    `seq<'a>`
    *opis*: Zwraca pustą kolekcję określonego typu

18. `List.exactlyOne lista`
    `Seq.exactlyOne źródło`
    *sygnatura*: `'a list->'a`
    `seq<'a>->'a`
    *opis*: Zwraca jedyny element w kolekcji. Jeżeli kolekcja nie zawiera dokładnie jednego elementu rzuca wyjątek

19. `List.except elementyDoWykluczenia lista`
    `Seq.except elementyDoWykluczenia źródło`
    *sygnatura*: `seq<'a>->'a list->'a list`
    `seq<'a>->seq<'a>->seq<'a>`
    *opis*: Zwraca nową kolekcję zawierającą unikalne wartości z kolekcji wejściowej, które nie pojawiają się w sekwencji elementyDoWykluczenia

20. `List.exists predykat lista`
    `Seq.exists predykat źródło`
    *sygnatura*: `'a->bool->'a list->bool`
    `'a->bool->seq<'a>->bool`
    *opis*: Określa czy jakikolwiek element kolekcji spełnia wymagania podanego predykatu

21. `List.exists2 predykat lista1 lista2`
    `List.exists2 predykat źródło1 źródło2`
    *sygnatura*: `'a->'b->bool->'a list->'b list->bool`
    `'a->'b->bool->seq<'a>->seq<'b>->bool`
    *opis*: Sprawdza, czy jakakolwiek para odpowiadających sobie elementów kolekcji spełnia podany predykat. Predykat jest stosowany dopóki nie skończy się mniejsza z obu kolekcji. Jeżeli sprawdzane elementy zwrócą `true`, to wynik funkcji jest `true` i dalsze sprawdzenia nie są dokonywane. Jeżeli jedna z kolekcji jest dłuższa to rzucany jest wyjątek, jeżeli nie to zwracana jest wartość `false`

22. `List.filter predykat lista`
    `Seq.filter predykat źródło`
    *sygnatura*: `'a->bool->'a list->'a list`
    `'a->bool->seq<'a>->seq<'a>`
    *opis*: Zwraca nową kolekcję zawierającą tylko te elementy kolekcji wejściowej, które spełniają podany predykat

23. `List.find predykat lista`
    `Seq.find predykat kolekcja`
    *sygnatura*: `'a->bool->'a list->'a`
    `'a->bool->seq<'a>->'a`
    *opis*: Zwraca pierwszy element, dla którego podana funkcja zwróci wartość `true`. Jeżeli nie ma takiego elementu, rzuca wyjątek

24. **`List.findBack predykat lista`**
    **`Seq.findBack predykat źródło`**
    *sygnatura*: `'a->bool->'a list->'a`
    `'a->bool->seq<'a>->'a`
    *opis*: Zwraca ostatni element, dla którego podana funkcja zwróci wartość `true` . Jeżeli nie ma takiego elementu, rzuca wyjątek. W przypadku sekwencji nie powinna być używana w przypadku dużych lub nieskończonych sekwencji wejściowych.

25. **`List.findIndex predykat lista`**
    **`Seq.findIndex predykat źródło`**
    *sygnatura*: `'a->bool->'a list->'a`
    `'a->bool->seq<'a>->'a`
    *opis*: Zwraca indeks pierwszego elementu, dla którego podana funkcja zwróci wartość `true`. Jeżeli nie ma takiego elementu, rzuca wyjątek

26. **`List.findIndexBack predykat lista`**
    **`Seq.findIndexBack predykat kolekcja`**
    *sygnatura*: `'a->bool->seq<'a>->'a`
    *opis*: Zwraca indeks ostatniego elementu, dla którego podana funkcja zwróci wartość `true`. Jeżeli nie ma takiego elementu, rzuca wyjątek. W przypadku sekwencji nie powinna być używana w przypadku dużych lub nieskończonych sekwencji wejściowych.

27. **`List.fold funkcja stan lista`**
    **`Seq.fold funkcja stan źródło`**
    *sygnatura*: `('s->'a->'s)->'s->'a list->'s`
    `('s->'a->'s)->'s->seq<'a>->'s`
    *opis*: Funkcja agregująca, przyjmująca jako początkową wartość `stan`. Agregacja jest realizowana funkcją `funkcja`

28. **`List.fold2 funkcja stan lista1 lista2`**
    **`Seq.fold2 funkcja stan źródło1 źródło2`**
    *sygnatura*: `('s->'a->'b->'s)->'s->'a list->'b list->'s`
    `('s->'a->'b->'s)->'s->seq<'a>->seq<'b>->'s`
    *opis*: Funkcja agregująca, przyjmująca jako początkową wartość `stan`. Agregacja jest realizowana funkcją `funkcja`, która jest stosowana do odpowiadających sobie elementów z obu kolekcji

29. **`List.foldBack funkcja lista stan`**
    **`Seq.foldBack funkcja źródło stan`**
    *sygnatura*: `('a->'s->'s)->'a list->'s->'s`
    `('a->'s->'s)->seq<'a>->'s->'s`
    *opis*: Funkcja agregująca, przyjmująca jako początkową wartość `stan`. Agregacja jest realizowana funkcją `funkcja`. Kolekcja przeglądana jest od końca do początku. W przypadku sekwencji nie powinna być używana w przypadku dużych lub nieskończonych sekwencji wejściowych.

30. `List.foldBack2 funkcja lista1 lista2 stan`
    `Seq.foldBack2 funkcja źródło1 źródło2 stan`
    *sygnatura*: `('a->'b->'s->'s)->'a list->'b list->'s->'s`
    `('a->'b->'s->'s)->seq<'a>->seq<'b>->'s->'s`
    *opis*: Funkcja agregująca, przyjmująca jako początkową wartość `stan`. Agregacja jest realizowana funkcją `funkcja`, która jest stosowana do odpowiadających sobie elementów z obu kolekcji. W przypadku sekwencji nie powinna być używana w przypadku dużych lub nieskończonych sekwencji wejściowych.

31. `List.forAll predykat lista`
    `Seq.forAll predykat kolekcja`
    *sygnatura*: `'a->bool->'a list->bool`
    `'a->bool->seq<'a>->bool`
    *opis*: Funkcja sprawdza, czy wszystkie elementy kolekcji spełniają warunek określony predykatem. Jeżeli predykat zwróci wartość `false` poszukiwanie jest przerywane

32. `List.forAll2 predykat lista1 lista2`
    `Seq.forAll2 predykat źródło1 źródło2`
    *sygnatura*: `'a->'b->bool->'a list->'b list->bool`
    `'a->'b->bool->seq<'a>->seq<'b>->bool`
    *opis*: Funkcja sprawdza, czy wszystkie odpowiadające sobie elementy obu kolekcji spełniają warunek określony predykatem. Predykat jest stosowany dopóki nie osiągnie końca krótszej kolekcji. Jeżeli predykat zwróci wartość `false` poszukiwanie jest przerywane. W przeciwnym przypadku jeżeli jedna kolekcja jest dłuższa, rzucany jest wyjątek. Inaczej zwracana jest wartość `true`

33. `List.groupBy projektor lista`
    `Seq.groupBy projektor kolekcja`
    *sygnatura*: `'a->'key->'a list->('key*'a list) list`
    `'a->'key->seq<'a>->seq<('key*seq<'a>)>`
    *opis*: Aplikuje funkcję generującą klucze do każdego elementu kolekcji tworząc kolekcję unikalnych kluczy. Do każdego klucza dołączona jest kolekcja elementów pasujących do tego klucza

34. `List.head lista`
    `Seq.head źródło`
    *sygnatura*: `'a list->'a`
    `seq<'a>->'a`
    *opis*: Zwraca pierwszy element kolekcji. Jeżeli kolekcja jest pusta rzuca wyjątek

35. `List.indexed lista`
    `Seq.indexed źródło`
    *sygnatura*: `'a list->(int*'a) list`
    `seq<'a> ->seq<(int*'a)>`
    *opis*: Zwraca nową kolekcję, której elementy odpowiadają elementom kolekcji wejściowej połączonych z ich indeksem

36. **`List.init długość inicjalizator`**
    **`Seq.init długość inicjalizator`**
    *sygnatura*: `int->int->'a->'a list`
    `int->int->'a->seq<'a>`
    *opis*: Tworzy kolekcję poprzez aplikowanie funkcji inicjalizującej do indeksu elementu
37. **`List.isEmpty lista`**
    **`Seq.isEmpty źródło`**
    *sygnatura*: `'a list->bool`
    `seq<'a>->bool`
    *opis*: Zwraca wartość `true`, gdy kolekcja nie zawiera elementów. W przeciwnym przypadku zwraca `false`
38. **`List.item index lista`**
    **`Seq.item index źródło`**
    *sygnatura*: `int->seq<'a>->'a`
    *opis*: Funkcja zwraca element o indeksie `index`. Rzuca wyjątek jeżeli atrubut `index` jest większy niż długość listy
39. **`List.iter akcja lista`**
    **`Seq.iter akcja źródło`**
    *sygnatura*: `'a->()->'a list->()`
    `'a->()->seq<'a>->()`
    *opis*: Aplikuje podaną funkcję do każdego elementu kolekcji
40. **`List.iter2 akcja lista1 lista2`**
    **`Seq.iter2 akcja źródło1 źródło2`**
    *sygnatura*: `'a->'b->()->seq<'a>->seq<'b>->()`
    *opis*: Aplikuje podaną funkcję do odpowiadających sobie elementów kolekcji. Obie kolekcje muszą mieć ten sam rozmiar
41. **`List.iteri akcja lista`**
    **`Seq.iteri akcja źródło`**
    *sygnatura*: `int->'a->()->'a list->()`
    `int->'a->()->seq<'a>->()`
    *opis*: Aplikuje podaną funkcję do każdego elementu kolekcji. Dodatkowo do funkcji przekazywany jest indeks elementu
42. **`List.iteri2 akcja lista1 lista2`**
    **`Seq.iteri2 akcja źródło1 źródło2`**
    *sygnatura*: `int->'a->'b->()->'a list->'b list->()`
    `int->'a->'b->()->seq<'a>->seq<'b>->()`
    *opis*: Aplikuje podaną funkcję do odpowiadających sobie elementów kolekcji. Dodatkowo do funkcji przekazywany jest indeks elementu. Obie kolekcje muszą mieć ten sam rozmiar

43. `List.last lista`
    `Seq.last źródło`
    *sygnatura*: `'a list->'a`
    `seq<'a>->'a`
    *opis*: Zwraca ostatni element kolekcji. Jeżeli kolekcja jest pusta rzuca wyjątek

44. `List.length lista`
    `Seq.length źródło`
    *sygnatura*: `'a list->int`
    `seq<'a>->int`
    *opis*: Zwraca liczbę elementów kolekcji.

45. `List.map mapa lista`
    `Seq.map mapa źródło`
    *sygnatura*: `'a->'b->'a list->'b list`
    `'a->'b->seq<'a>->seq<'b>`
    *opis*: Tworzy nową kolekcję poprzez zastosowanie funkcji do każdego elementu kolekcji wejściowej

46. `List.map2 mapa lista1 lista2`
    `Seq.map2 mapa źródło1 źródło2`
    *sygnatura*: `'a->'b->'c->'a list->'b list->'c list`
    `'a->'b->'c->seq<'a>->seq<'b>->seq<'c>`
    *opis*: Tworzy nową kolekcję poprzez zastosowanie funkcji do odpowiadających sobie elementów kolekcji wejściowych.

47. `List.map3 mapa lista1 lista2 lista3`
    `Seq.map3 mapa źródło1 źródło2 źródło3`
    *sygnatura*: `'a->'b->'c->'d->'a list->'b list->'c list->'d list`
    `'a->'b->'c->'d->seq<'a>->seq<'b>->seq<'c>->seq<'d>`
    *opis*: Tworzy nową kolekcję poprzez zastosowanie funkcji do odpowiadających sobie elementów kolekcji wejściowych.

48. `List.mapFold mapa stan lista`
    `Seq.mapFold mapa stan źródło`
    *sygnatura*: `'stan->'a->('b*'stan)->'stan->'a list->'b list*'stan`
    `'stan->'a->('b*'stan)->'stan->seq<'a>->seq<'b>*'stan`
    *opis*: Łączy operacje map i fold. Zwraca nową kolekcję jak map i zakumulowaną wartość jak w fold

49. `List.mapFoldBack mapa lista stan`
    `Seq.mapFoldBack mapa kolekcja stan`
    *sygnatura*: `'a->'stan->('b*'stan)->'a list->'stan->'b list*'stan`
    `'a->'stan->('b*'stan)->seq<'a>->'stan->seq<'b>*'stan`
    *opis*: Łączy operacje map i fold. Zwraca nową kolekcję jak map i zakumulowaną wartość jak w fold. Przegląda kolekcję od końca. W przypadku sekwencji nie powinna być używana w przypadku dużych lub nieskończonych sekwencji wejściowych.

50. `List.mapi mapa lista`
    `Seq.mapi mapa źródło`
    *sygnatura*: `int->'a->'b->'a list->'b list`
    `int->'a->'b->seq<'a>->seq<'b>`
    *opis*: Tworzy nową kolekcję poprzez zastosowanie funkcji do każdego elementu kolekcji wejściowej. Dodatkowo do funkcji przekazuje również indeks elementu
51. `List.mapi2 mapa lista1 lista2`
    `Seq.mapi2 mapa źródło1 źródło2`
    *sygnatura*: `int->'a->'b->'c->'a list->'b list->'c list`
    `int->'a->'b->'c->seq<'a> ->seq<'b>->seq<'c>`
    *opis*: Tworzy nową kolekcję poprzez zastosowanie funkcji do odpowiadających sobie elementów kolekcji wejściowych. Dodatkowo do funkcji przekazuje również indeks elementu
52. `List.max lista`
    `Seq.max źródło`
    *sygnatura*: `'a list->'a`
    `seq<'a>->'a`
    *opis*: Zwraca największy element kolekcji. Elementy sa porównywane za pomocą operatora max
53. `List.maxBy projektor lista`
    `Seq.maxBy projektor źródło`
    *sygnatura*: `'a->'b->'a list->'b`
    `'a->'b->seq<'a>->'b`
    *opis*: Aplikuje funkcję do każdego elementu kolekcji, a następnie wybiera największy z wyników. Elementy sa porównywane za pomocą operatora max
54. `List.min lista`
    `Seq.min źródło`
    *sygnatura*: `'a list->'a`
    `seq<'a>->'a`
    *opis*: Zwraca najmniejszy element kolekcji. Elementy są porównywane za pomocą operatora max
55. `List.minBy projektor lista`
    `Seq.minBy projektor źródło`
    *sygnatura*: `'a->'b->'a list->'b`
    `'a->'b->seq<'a>->'b`
    *opis*: Aplikuje funkcję do każdego elementu kolekcji, a następnie wybiera najmniejszy z wyników. Elementy są porównywane za pomocą operatora max
56. `List.ofArray tablica`
    `Seq.ofArray tablica`
    *sygnatura*: `'a[]->'a list`
    `'a[]->seq<'a>`
    *opis*: Tworzy listę lub sekwencję na podstawie tablicy

57. `Seq.ofList lista`
    *sygnatura*: `'a list->seq<'a>`
    *opis*: Tworzy sekwencję na podstawie listy

58. `List.ofSeq seq`
    *sygnatura*: `seq<'a>->'a list`
    *opis*: Tworzy listę na podstawie sekwencji

59. `List.pairwise lista`
    `Seq.pairwise źródło`
    *sygnatura*: `'a list->('a*'a) list`
    `seq<'a>->seq<('a*'a)>`
    *opis*: Zwraca kolekcję par: element kolekcji i jego następnik.

60. `List.partition predicate lista`
    *sygnatura*: `'a->bool->'a list->('a list*'a list)`
    *opis*: Dzieli kolekcje na dwie części. Pierwsza zawiera elementy, dla których predykat zwraca wartość `true`, druga zawiera elementy dla których predykat zwraca wartość `false`. Kolejność elementów jest zachowywana

61. `List.permute mapaIndeksow lista`
    `Seq.permute mapaIndeksow źródło`
    *sygnatura*: `int->int->'a list->'a list`
    `int->int->seq<'a>->seq<'a>`
    *opis*: Zwraca nową kolekcję, która zawiera permutację elementów wejściowych. Permutacja jest określona za pomocą funkcji `mapaIndeksow`

62. `List.pick wyborca lista`
    `Seq.pick wyborca źródło`
    *sygnatura*: `'a->'b option->'a list->'b`
    `'a->'b option->seq<'a>->'b`
    *opis*: Aplikuje funkcję do kolejnych elementów kolekcji. Zwraca ten element `x`, dla której funkcja osiąga wartość `Some(x)`. Jeżeli takiego elementu nie ma rzuca wyjątek

63. `List.reduce reduktor lista`
    `Seq.reduce reduktor źródło`
    *sygnatura*: `'a->'a->'a->'a list->'a`
    `'a->'a->'a->seq<'a>->'a`
    *opis*: Aplikuje funkcję do dwóch pierwszych elementów kolekcji. Następnie uzyskany wynik przekazuje do funkcji razem z trzecim elementem itd. Po osiągnięciu ostatniego elementu kolekcji zwraca ostateczny rezultat.

64. `List.reduceBack reduktor lista`
    `Seq.reduceBack reduktor źródło`
    *sygnatura*: `'a->'a->'a->'a list->'a`
    `'a->'a->'a->seq<'a>->'a`
    *opis*: Aplikuje funkcję do dwóch pierwszych elementów kolekcji. Następnie uzyskany wynik przekazuje do funkcji razem z trzecim elementem itd. Po osiągnięciu ostatniego elementu kolekcji zwraca ostateczny rezultat. Przegląda kolekcję od końca. W przypadku sekwencji nie powinna być używana w przypadku dużych lub nieskończonych sekwencji wejściowych.
65. `List.replicate n incjalizator`
    `Seq.replicate n incjalizator`
    *sygnatura*: `int->'a->'a list`
    `int->'a->seq<'a>`
    *opis*: Tworzy nową kolekcję poprzez n-krotne sklonowanie elementu incjalizator
66. `List.rev lista`
    `Seq.rev źródło`
    *sygnatura*: `'a list->'a list`
    `seq<'a>->seq<'a>`
    *opis*: Zwraca nową kolekcję z elementami w odwrotnym porządku. Uwaga! w przypadku sekwencji funkcja najpierw pobierze całą sekwencję wejściową zanim zwróci pierwszą wartość
67. `List.scan funkcja stan lista`
    `Seq.scan funkcja stan źródło`
    *sygnatura*: `'a->'b->'a->'a->'b list->'a list`
    `'a->'b->'a->'a->seq<'b>->seq<'a>`
    *opis*: Działa jak fold, ale zwraca nie ostateczny wynik, a kolekcję wyników pośrednich
68. `List.scanBack funkcja lista stan`
    `Seq.scanBack funkcja stan lista`
    *sygnatura*: `'b->'a->'a->'b list->'a->'a list`
    `'b->'a->'a->seq<'b>->'a->seq<'a>`
    *opis*: Działa jak foldBack, ale zwraca nie ostateczny wynik, a kolekcję wyników pośrednich. W przypadku sekwencji nie powinna być używana w przypadku dużych lub nieskończonych sekwencji wejściowych.
69. `List.singleton wartość`
    `Seq.singleton wartość`
    *sygnatura*: `'a->'a list`
    `'a->seq<'a>`
    *opis*: Zwraca kolekcję zawierającą jeden element

70. `List.skip n lista`
    `Seq.skip n źródło`
    *sygnatura*: `int->'a list->'a list`
    `int->seq<'a>->seq<'a>`
    *opis*: Zwraca nową kolekcję bez n pierwszych elementów. Jeżeli n jest ujemne lub większe niż liczba elementów kolekcji rzuca wyjątek

71. `List.skipWhile predykat lista`
    `Seq.skipWhile predykat źródło`
    *sygnatura*: `'a->bool->'a list->'a list`
    `'a->bool->seq<'a>->seq<'a>`
    *opis*: Pomija elementy kolekcji dopóki predykat zwraca wartość `true`. Zwraca pozostałe elementy kolekcji

72. `List.sort lista`
    `Seq.sort źródło`
    *sygnatura*: `'a list->'a list`
    `seq<'a>->seq<'a>`
    *opis*: Sortuje elementy kolekcji, porównując je za pomocą operatora compare. W przypadku sekwencji nie powinna być używana w przypadku dużych lub nieskończonych sekwencji wejściowych.

73. `List.sortBy projektor lista`
    `Seq.sortBy projektor źródło`
    *sygnatura*: `'a->'key->'a list->'a list`
    `'a->'key->seq<'a>->seq<'a>`
    *opis*: Sortuje kolekcję używając kluczy wygenerowanych przez projektor. Klucze są porównywane za pomocą operatora compare. W przypadku sekwencji nie powinna być używana w przypadku dużych lub nieskończonych sekwencji wejściowych.

74. `List.sortByDescending projektor lista`
    `Seq.sortByDescending projektor źródło`
    *sygnatura*: `'a->'key->seq<'a>->seq<'a>`
    *opis*: Sortuje kolekcję w kolejności malejącej używając kluczy wygenerowanych przez projektor. Klucze są porównywane za pomocą operatora compare. W przypadku sekwencji nie powinna być używana w przypadku dużych lub nieskończonych sekwencji wejściowych.

75. `List.sortDescending lista`
    `Seq.sortDescending źródło`
    *sygnatura*: `'a list->'a list`
    `seq<'a>->seq<'a>`
    *opis*: Sortuje elementy kolekcji w kolejności malejącej, porównując je za pomocą operatora compare. W przypadku sekwencji nie powinna być używana w przypadku dużych lub nieskończonych sekwencji wejściowych.

76. **`List.sortWith komparator lista`**
    **`Seq.sortWith komparator źródło`**
    *sygnatura*: `'a->'a->int->'a list->'a list`
    `'a->'a->int->seq<'a>->seq<'a>`
    *opis*: Sortuje kolekcję wykorzystując funkcję komparator do porównywania elementów. W przypadku sekwencji nie powinna być używana w przypadku dużych lub nieskończonych sekwencji wejściowych.

77. **`List.splitAt indeks lista`**
    *sygnatura*: `int->'a list->'a list*'a list`
    *opis*: Funkcja dzieli listę na dwie częście względem podanego indeksu

78. **`List.splitInto n lista`**
    **`Seq.splitInto n źródło`**
    *sygnatura*: `int->'a list->'a list list`
    `int->seq<'a>->seq<'a[]>`
    *opis*: Dzieli kolekcję na co najwyżej n fragmentów

79. **`List.sum lista`**
    **`Seq.sum źródło`**
    *sygnatura*: `^a list->^a`
    `seq<^a> ->^a`
    *opis*: Funkcja oblicza sumę elementów kolekcji

80. **`List.sumBy projektor lista`**
    **`Seq.sumBy projektor źródło`**
    *sygnatura*: `'a->^b->'a list->^b`
    `'a->^b->seq<'a>->^b`
    *opis*: Funkcja oblicza sumę elementów kolekcji, do których zastosowano funkcję projektor

81. **`List.tail lista`**
    **`Seq.tail źródło`**
    *sygnatura*: `'a list->'a list`
    `seq<'a>->seq<'a>`
    *opis*: Zwraca kolekcję bez pierwszego elementu

82. **`List.take n lista`**
    **`Seq.take n źródło`**
    *sygnatura*: `int->'a list->'a list`
    `int->seq<'a>->seq<'a>`
    *opis*: Zwraca n pierwszych elementów kolekcji

83. `List.takeWhile predykat lista`
    `Seq.takeWhile predykat źródło`
    *sygnatura*: `'a->bool->'a list->'a list`
    `'a->bool->seq<'a>->seq<'a>`
    *opis*: Pobiera elementy z kolekcji dopóki pradykat zwraca wartość `true`. Odrzuca pozostałe.
84. `List.toArray lista`
    `Seq.toArray źródło`
    *sygnatura*: `'a list->'a[]`
    `seq<'a>->'a[]`
    *opis*: Tworzy tablicę na podstawie kolekcji wejściowej

85. `Seq.toList źródło`
    *sygnatura*: `seq<'a>->'a list`
    *opis*: Tworzy listę na podstawie sekwencji

86. `List.toSeq lista`
    *sygnatura*: `'a list->seq<'a>`
    *opis*: Tworzy sekwencje na podstawie listy

87. `List.transpose listy`
    `Seq.transpose źródło`
    *sygnatura*: `seq<'a list>->'a list list`
    `seq<kolekcja>->seq<seq<'a»`
    *opis*: Transponuje podaną sekwencję kolekcji

88. `List.truncate n lista`
    `Seq.truncate n źródło`
    *sygnatura*: `int->'a list->'a list`
    `int->seq<'a>->seq<'a>`
    *opis*: Zwraca co najwyżej n pierwszych elementów kolekcji

89. `List.tryExactlyOne lista`
    `Seq.tryExactlyOne źródło`
    *sygnatura*: `'a list->'a option`
    `seq<'a>->'a option`
    *opis*: Zwraca jedyny element w kolekcji. Jeżeli lista nie zawiera dokładnie jednego elementu zwraca None
90. `List.tryFind predykat lista`
    `Seq.tryFind predykat źródło`
    *sygnatura*: `'a->bool->'a list->'a option`
    `'a->bool->seq<'a>->'a option`
    *opis*: Zwraca pierwszy element, dla którego podana funkcja zwróci wartość `true`. Jeżeli nie ma takiego elementu, zwraca None

91. **`List.tryFindBack predykat lista`**
    **`Seq.tryFindBack predykat źródło`**
    *sygnatura*: `'a->bool->seq<'a>->'a option`
    *opis*: Zwraca ostatni element, dla którego podana funkcja zwróci wartość `true` . Jeżeli nie ma takiego elementu, zwraca None. W przypadku sekwencji nie powinna być używana w przypadku dużych lub nieskończonych sekwencji wejściowych.

92. **`List.tryFindIndex predykat lista`**
    **`Seq.tryFindIndex predykat źródło`**
    *sygnatura*: `'a->bool->'a list->'a option`
    `'a->bool->seq<'a>->'a option`
    *opis*: Zwraca indeks pierwszego elementu, dla którego podana funkcja zwróci wartość `true`. Jeżeli nie ma takiego elementu, zwraca None

93. **`List.tryFindIndexBack predykat lista`**
    **`Seq.tryFindIndexBack predykat źródło`**
    *sygnatura*: `'a->bool->'a list->'a option`
    `'a->bool->seq<'a>->'a option`
    *opis*: Zwraca indeks ostatniego elementu, dla którego podana funkcja zwróci wartość `true`. Jeżeli nie ma takiego elementu, zwraca None. W przypadku sekwencji nie powinna być używana w przypadku dużych lub nieskończonych sekwencji wejściowych.

94. **`List.tryHead lista`**
    **`Seq.tryHead źródło`**
    *sygnatura*: `seq<'a>->'a option`
    *opis*: Zwraca pierwszy element kolekcji. Jeżeli kolekcja jest pusta zwraca None

95. **`List.tryItem index lista`**
    **`Seq.tryItem index kolekcja`**
    *sygnatura*: `int->'a list->'a option`
    `int->seq<'a>->'a option`
    *opis*: Funkcja zwraca element o indeksie `index`. Zwraca None jeżeli atrubut `index` jest większy niż długość kolekcji

96. **`List.tryLast lista`**
    **`Seq.tryLast źródło`**
    *sygnatura*: `'a list->'a option`
    `seq<'a>->'a option`
    *opis*: Zwraca ostatni element kolekcji. Jeżeli kolekcja jest pusta zwraca None

97. **`List.tryPick wyborca lista`**
    **`Seq.tryPick wyborca źródło`**
    *sygnatura*: `'a->'b option->'a list->'b option`
    `'a->'b option->seq<'a>->'b option`
    *opis*: Aplikuje funkcję do kolejnych elementów kolekcji. Zwraca ten element `x`, dla której funkcja osiąga wartość `Some(x)`. Jeżeli takiego elementu nie ma zwraca None

98. `List.unfold generator stan`
    `Seq.unfold generator stan`
    *sygnatura*: `'a->('b*'a)->'a->'a list`
    `'a->('b*'a)->'a->seq<'a>`
    *opis*: Zwraca kolekcję, która zawiera elementy wygenerowane przez generator. Generowanie kolejnych elementów może być zależne od przekazanego stanu
99. `List.unzip lista`
    *sygnatura*: `('a*'b) list->'a list * 'b list`
    *opis*: Rozdziela listę par na dwie osobne listy

100. `List.unzip3 lista`
    *sygnatura*: `('a*'b*'c) list->'a list * 'b list * 'c list`
    *opis*: Rozdziela listę trójek na trzy osobne listy

101. `List.where predykat lista`
    `Seq.where predykat źródło`
    *sygnatura*: `'a->bool->'a list->'a list`
    `'a->bool->seq<'a>->seq<'a>`
    *opis*: Zwraca nową kolekcję zawierającą tylko te elementy dla których predykat zwraca wartość `true`
102. `List.windowed rozmiarOkna lista`
    `Seq.windowed rozmiarOkna źródło`
    *sygnatura*: `int->seq<'a>->seq<'a[]>`
    *opis*: Zwraca kolekcję przesuwanych okien zawierających elementy pobrane z kolekcji wejściowej. Każde okno zwracane jest jako osobna kolekcja
103. `List.zip lista1 lista2`
    `Seq.zip źródło1 źródło2`
    *sygnatura*: `'a list->'b list->('a*'b) list`
    `seq<'a>->seq<'b>->seq<('a*'b)>`
    *opis*: Łączy dwie kolekcje w kolekcję par. Obie kolekcje muszą być tej samej długości
104. `List.zip3 lista1 lista2 lista3`
    `Seq.zip3 źródło1 źródło2 źródło3`
    *sygnatura*: `'a list->'b list->'c list->('a*'b*'c) list`
    `seq<'a>->seq<'b>->seq<'c>->seq<('a*'b*'c)>`
    *opis*: Łączy trzy kolekcje w kolekcję trójek. Wszystkie kolekcje muszą być tej samej długości

## A.2 Zbiory - moduł Map

1. `Map.add klucz wartosc mapa`
    *sygnatura*: `'a->'b->Map<'a,'b>->Map<'a,'b>`
    *opis*: Tworzy nową mapę, która dodatkowo zawiera powiązania klucza z wartością

2. **`Map.change klucz funkcja mapa`**
   *sygnatura*: `'a->'b option->'b option->Map<'a,'b>->Map<'a,'b>`
   *opis*: Zwraca nową mapę, która zmienia wartość przypisaną do klucza zgodnie z funkcją
3. **`Map.containsKey klucz mapa`**
   *sygnatura*: `'a->Map<'a,'b>->bool`
   *opis*: Sprawdza, czy dany klucz jest obecny w mapie
4. **`Map.count mapa`**
   *sygnatura*: `Map<'a,'b>->int`
   *opis*: Zwraca liczbę elementów w mapie
5. **`Map.empty`**
   *sygnatura*: `Map<'a,'b>`
   *opis*: Zwraca pustą mapę
6. **`Map.exists predykat mapa`**
   *sygnatura*: `'a->'b->bool->Map<'a,'b>->bool`
   *opis*: Sprawdza czy podany predykat zwraca wartość `true` dla co najmniej jednego elementu mapy
7. **`Map.filter predykat mapa`**
   *sygnatura*: `'a->'b->bool->Map<'a,'b>->Map<'a,'b>`
   *opis*: Tworzy nową mapę, która zawiera tylko te elementy dla których podany predykat zwraca wartość `true`
8. **`Map.find klucz mapa`**
   *sygnatura*: `'a->Map<'a,'b>->'b`
   *opis*: Szuka elementu w mapie na podstawie klucza. Jeżeli element nie istnieje w mapie, rzucany jest wyjątek
9. **`Map.findKey predykat mapa`**
   *sygnatura*: `'a->'b->bool->Map<'a,'b>->'a`
   *opis*: Zwraca klucz dla pierwszego elementu mapy, dla którego predykat zwraca wartość `true`
10. **`Map.fold funkcja stan mapa`**
    *sygnatura*: `'stan->'a->'b->'stan->'stan->Map<'a,'b>->'stan`
    *opis*: Wykonuje operację fold dla wszystkich elementów mapy
11. **`Map.foldBack funkcja mapa stan`**
    *sygnatura*: `'a->'b->'stan->'stan->Map<'a,'b>->'stan->'stan`
    *opis*: Wykonuje operację fold dla wszystkich elementów mapy. Przegląda mapę w odwrotnej kolejności

12. `Map.forall predykat mapa`
    *sygnatura*: `'a->'b->bool->Map<'a,'b>->bool`
    *opis*: Sprawdza czy podany predykat zwraca wartość `true` dla wszystkich elementów mapy
13. `Map.isEmpty mapa`
    *sygnatura*: `Map<'a,'b>->bool`
    *opis*: Sprawdza czy mapa jest pusta
14. `Map.iter akcja mapa`
    *sygnatura*: `'a->'b->unit->Map<'a,'b>->unit`
    *opis*: Stosuje podaną funkcję do każdego elementu mapy
15. `Map.map funkcja mapa`
    *sygnatura*: `'a->'b->'u->Map<'a,'b>->Map<'a,'u>`
    *opis*: Tworzy nową mapę, której elementami są wartości zwrócone przez podaną funkcję zastosowaną do wszystkich elementów mapy wejściowej
16. `Map.ofArray tablica`
    *sygnatura*: `('a*'b)[]->Map<'a,'b>`
    *opis*: Zwraca nową mapę stworzoną na podstawie tablicy
17. `Map.ofList lista`
    *sygnatura*: `('a*'b) list->Map<'a,'b>`
    *opis*: Zwraca nową mapę stworzoną na podstawie listy
18. `Map.ofSeq kolekcja`
    *sygnatura*: `seq<('a*'b)>->Map<'a,'b>`
    *opis*: Zwraca nową mapę stworzoną na podstawie sekwencji
19. `Map.partition predykat mapa`
    *sygnatura*: `'a->'b->bool->Map<'a,'b>->Map<'a,'b>*Map<'a,'b>`
    *opis*: Tworzy dwie mapy. Pierwsza zawiera te elementy mapy wejściowej, dla której predykat zwraca wartość `true`, druga te elementy dla której funkcja zwróciła `false`
20. `Map.pick wyborca mapa`
    *sygnatura*: `'a->'b->'u option->Map<'a,'b>->'u`
    *opis*: Przeszukuje mapę w poszukiwaniu pierwszego elementu, dla którego podana funkcja zwróci wartość `Some(x)`
21. `Map.remove klucz mapa`
    *sygnatura*: `'a->Map<'a,'b>->Map<'a,'b>`
    *opis*: Usuwa element o podanym kluczu z mapy

22. `Map.toArray mapa`
    *sygnatura*: `Map<'a,'b>->('a*'b)[]`
    *opis*: Zwraca tablicę zawierającą pary klucz,wartość znajdujące się w mapie

23. `Map.toList mapa`
    *sygnatura*: `Map<'a,'b>->('a*'b) list`
    *opis*: Zwraca listę zawierającą pary klucz,wartość znajdujące się w mapie

24. `Map.toSeq mapa`
    *sygnatura*: `Map<'a,'b>->seq<('a*'b)>`
    *opis*: Zwraca sekwencję zawierającą pary klucz,wartość znajdujące się w mapie

25. `Map.tryFind klucz mapa`
    *sygnatura*: `'a->Map<'a,'b>->'a option`
    *opis*: Szuka w mapie elementu o podanym kluczu. Jeżeli go znajdzie zwraca wartość `Some<x>`, jeżeli nie - zwraca `None`

26. `Map.tryFindKey predykat mapa`
    *sygnatura*: `'a->'b->bool->Map<'a,'b>->'a option`
    *opis*: Zwraca klucz dla pierwszego elementu mapy, dla którego predykat zwraca wartość `true`. Jeżeli zostanie taki znaleziony zwraca wartość `Some(klucz)`, jeżeli nie - zwraca `None`

27. `Map.tryPick wyborca mapa`
    *sygnatura*: `'a->'b->'u option->Map<'a,'b>->'u option`
    *opis*: Przeszukuje mapę w poszukiwaniu pierwszego elementu, dla którego podana funkcja zwróci wartość `Some(x)`. Jeżeli go znajdzie zwraca wartość `Some<x>`, jeżeli nie - zwraca `None`

## A.3 Zbiory - moduł Set

1. `Set.add wartosc zbiór`
    *sygnatura*: `'a->Set<'a>->Set<'a>`
    *opis*: Tworzy nowy zbiór z dołączonym nowym elementem

2. `Set.contains wartość zbiór`
    *sygnatura*: `'a->Set<'a>->bool`
    *opis*: Sprawdza, czy dana wartość jest obecna w zbiorze

3. `Set.count zbiór`
    *sygnatura*: `Set<'a>->int`
    *opis*: Zwraca liczbę elementów w zbiorze

4. `Set.difference zbiór1 zbiór2`
   *sygnatura*: `Set<'a>->Set<'a>->Set<'a>`
   *opis*: Zwraca nowy zbiór zawierający te elementu zbioru pierwszego, które nie występują w drugim
5. `Set.empty`
   *sygnatura*: `Set<'a>`
   *opis*: Zwraca pusty zbiór
6. `Set.exists predykat zbiór`
   *sygnatura*: `'a->bool->Set<'a>->bool`
   *opis*: Sprawdza czy podany predykat zwraca wartość `true` dla co najmniej jednego elementu zbioru
7. `Set.filter predykat zbiór`
   *sygnatura*: `'a->bool->Set<'a>->Set<'a>`
   *opis*: Tworzy nowy zbiór, która zawiera tylko te elementy dla których podany predykat zwraca wartość `true`
8. `Set.fold funkcja stan zbiór`
   *sygnatura*: `'stan->'a->'stan->'stan->Set<'a>->'stan`
   *opis*: Wykonuje operację fold dla wszystkich elementów zbioru
9. `Set.foldBack funkcja zbiór stan`
   *sygnatura*: `'a->'stan->'stan->Set<'a>->'stan->'stan`
   *opis*: Wykonuje operację fold dla wszystkich elementów zbioru. Przegląda zbiór w odwrotnej kolejności
10. `Set.forall predykat zbiór`
    *sygnatura*: `'a->bool->Set<'a>->bool`
    *opis*: Sprawdza czy podany predykat zwraca wartość `true` dla wszystkich elementów zbioru
11. `Set.intersect zbiór1 zbiór2`
    *sygnatura*: `Set<'a>->Set<'a>->Set<'a>`
    *opis*: Oblicza przecięcie dwóch zbiorów
12. `Set.intersectMany zbiory`
    *sygnatura*: `seq<Set<'a»->Set<'a>`
    *opis*: Oblicza przecięcie kolekcji zbiorów. Kolekcja nie może być pusta
13. `Set.isEmpty zbiór`
    *sygnatura*: `Set<'a>->bool`
    *opis*: Sprawdza czy zbiór jest pusty
14. `Set.isProperSubset zbiór1 zbiór2`
    *sygnatura*: `Set<'a>->Set<'a>->bool`
    *opis*: Sprawdza zbiór pierwszy jest poprawnym podzbiorem zbioru 2

15. `Set.isProperSuperset zbiór1 zbiór2`
    *sygnatura*: `Set<'a>->Set<'a>->bool`
    *opis*: Sprawdza zbiór pierwszy jest poprawnym nadzbiorem zbioru 2

16. `Set.isSubset zbiór1 zbiór2`
    *sygnatura*: `Set<'a>->Set<'a>->bool`
    *opis*: Sprawdza zbiór pierwszy jest podzbiorem zbioru 2

17. `Set.isSuperset zbiór1 zbiór2`
    *sygnatura*: `Set<'a>->Set<'a>->bool`
    *opis*: Sprawdza zbiór pierwszy jest nadzbiorem zbioru 2

18. `Set.iter akcja zbiór`
    *sygnatura*: `'a->unit->Set<'a>->unit`
    *opis*: Stosuje podaną funkcję do każdego elementu zbioru

19. `Set.map funkcja zbiór`
    *sygnatura*: `'a->'u->Set<'a>->Map<'a,'u>`
    *opis*: Tworzy nowy zbiór, którego elementami są wartości zwrócone przez podaną funkcję zastosowaną do wszystkich elementów zbioru wejściowego

20. `Set.maxElement zbiór`
    *sygnatura*: `Set<'a>->'a`
    *opis*: Zwraca największy element zbioru zgodnie z porządkiem jaki został użyty w zbiorze

21. `Set.minElement zbiór`
    *sygnatura*: `Set<'a>->'a`
    *opis*: Zwraca najmniejszy element zbioru zgodnie z porządkiem jaki został użyty w zbiorze

22. `Set.ofArray tablica`
    *sygnatura*: `'a[]->Set<'a>`
    *opis*: Zwraca nowy zbiór stworzony na podstawie tablicy

23. `Set.ofList lista`
    *sygnatura*: `'a list->Set<'a>`
    *opis*: Zwraca nowy zbiór stworzony na podstawie listy

24. `Set.ofSeq kolekcja`
    *sygnatura*: `seq<'a>->Set<'a>`
    *opis*: Zwraca nowy zbiór stworzony na podstawie sekwencji

25. `Set.partition predykat zbiór`
    *sygnatura*: `'a->'b->bool->Set<'a>->Set<'a>*Set<'a>`
    *opis*: Tworzy dwa zbiory. Pierwszy zawiera te elementy zbioru wejściowego, dla którego predykat zwraca wartość `true`, druga te elementy dla których funkcja zwróciła `false`
26. `Set.remove klucz zbiór`
    *sygnatura*: `'a->Set<'a>->Set<'a>`
    *opis*: Usuwa element ze zbioru

27. `Set.singleton wartość`
    *sygnatura*: `'a->Set<'a>`
    *opis*: Tworzy zbiór zawierający jeden element

28. `Set.toArray zbiór`
    *sygnatura*: `Set<'a>->'a[]`
    *opis*: Zwraca tablicę zawierającą elementy zbioru

29. `Set.toList zbiór`
    *sygnatura*: `Set<'a>->'a list`
    *opis*: Zwraca listę zawierającą elementy zbioru

30. `Set.toSeq zbiór`
    *sygnatura*: `Set<'a>->seq<'a>`
    *opis*: Zwraca sekwencję zawierającą elementy zbioru

31. `Set.union zbiór1 zbiór2`
    *sygnatura*: `Set<'a>->Set<'a>->Set<'a>`
    *opis*: Oblicza sumę dwóch zbiorów

32. `Set.unionMany zbiory`
    *sygnatura*: `seq<Set<'a»->Set<'a>`
    *opis*: Oblicza sumę kolekcji zbiorów. Kolekcja nie może być pusta