1. Napisati implementaciju metode `:::` koja vrši konkatenaciju dve generičke liste.
   `def :::[B](prefix: List[B], l: List[B]): List[B]`

2. Napisati implementaciju funkcije `take`, čiji rezultat je nova lista formirana od prvih `n` elemenata generičke liste. Ako je `n` veće od dužine liste, rezultat je sama lista.
   `def take[T](l : List[T], n : Int) : List[T]`

3. Napisati implementaciju funkcije `drop`, čiji rezultat je nova lista formirana izostavljajući prvih `n` elemenata generičke liste. Ako je `n` veće od dužine liste, rezultat je prazna lista.
   `def drop[T](l : List[T], n : Int) : List[T]`

4. Napisati implementaciju funkcije `splitAt`, čiji rezultat je par lista. Prvi element para je lista formirana izostavljajući prvih `n` elemenata ulazne liste. Drugi element para je lista koja se sastoji od preostalih elemenata ulazne liste.
   `def splitAt[T](l : List[T], n : Int) : (List[T], List[T])`

5. Implementirati metodu `apply` koja vraća `n`-ti element generičke liste.
   `def apply[T](l : List[T], n : Int) : T`

6. Napisati funkciju `map` koja od zadate generičke liste formira novu listu primenom funkcije preslikavanja na njene elemente.
   `def map[S, D](f : S => D)(l : List[S])`