Esercizio (dictionary)
Un dictionary \`e un tipo astratto per la memorizzazione e l'accesso di informazioni attraverso l'uso di chiavi. Ha le seguenti operazioni:
    add: aggiunge un dato v con chiave di accesso k, se k non presente, oppure
         \`e presente ma \`e unbound. Altrimenti ...
    get: restituisce il dato all'accesso k, se presente, se k definita.
         Altrimenti...
    remove: rimuove il dato associato alla chiave k, se presente, se k definita.
         Altrimenti...
    keys: restituisce la lista delle chiavi nel dictionary
    elms: restituisce la lista delle informazioni residenti nel dictionary
    size: restituisce il numero di chiavi introdotte (con dato, oppure no)
    isEmpty: true se e solo se tutte le chiavi definite sono unbound
In aggiunta \`e previsto un costruttore per dictionary vuoti. Chiavi e dati sono di tipo arbitrario. Si forniscano:
(a) API in Ocaml per IMMUTABLE e in Java per MUTABLE completando i comportamenti
    "..." lasciati in sospeso.
(b) Ocaml: ADT, AF, I, operazioni includenti equals, toString, copy
(c) Java: ADT, AF, I, operazioni includenti equals, toString, clone
(d) Esempi di uso, in OCaml e in Java, su dictionary con almeno 5 dati ed che
    usino tutte le operazioni di ciascun ADT definito.
(e) Estensioni di entrambi gli ADT al caso di dictionary a chiave singola, dove
    ogni valore v deve avere un'unica chiave di accesso.

Esercizio (Polinomio in Java)
Sia ${\tt c_{n}x^{n}+...+c_{0}}$,
con ${\tt n\!\geq\!0, c_{n}\neq\!0, (i\in[0,n])\,c_{i}\!\in\![int] un polinomio a coefficienti interi di grado n. Si definisca in Java un tipo astratto per polinomi a struttura e valore dei componenti MODIFICABILI.
In particolare:
 (a) Si definisca lo stato concreto;
 (b) Le funzioni AF ed I;
 (c) Si definiscano i costruttori;
 (d) Si definisca 1 operazione osservatore e, 1 operazione modificatore.
 (e) Si fornisca un caso di uso
 (f) Si forniscano gli opportuni additional (equals, clone, toString)
 (g) Si estenda il caso di uso agli additional

Esercizio (Polinomio in OCaml)
Sia ${\tt c_{n}x^{n}+...+c_{0}}$, un polinomio a coefficienti interi di grado n, come sopra. Si definisca in OCaml un tipo astratto poly per polinomi NON-MODIFICABILI, con le seguenti operazioni:
    mk: riceve una lista di coppie ciascuna indicante il coefficiente e il grado
        di ogni addendo del polinomio;
    isZero: riconosce il poly 0
    add: calcola la somma di polinomi
    sub: calcola la sottrazione
    mul: calcola il prodotto

```
    div: calcola la divisione se possibile ottenere un polinomio a coefficienti
         interi. In caso contrario solleva eccezione
    equals: riconosce polinomi uguali
    toSting: genera una stringa di presentazione del polinomio.
Si forniscano:
 (a) API
 (a) Stato concreto e funzioni AF ed I per un ADT;
 (d) Le operazioni elencate per tale ADT.
 (e) Un caso di uso per ogni operazione.

Esercizio (Relazione Binaria in Ocaml)
Sia rel2 un tipo astratto su generici tipi A e B, per relazioni binarie (finite)
della forma ${\tt \{(x_1,y_1),...,(x_n,y_n)\}}$, con gli ${\tt x_i}$ di tipo A,
e gli ${\tt y_i}$ di tipo B. Su questi valori sono definite le seguenti operazioni
    isempty: la relazione vuota
    add: aggiunge una coppia
    rem: rimuove una coppia
    get1: lista di tutti i primi componenti aventi l'argomento come secondo
    get2: lista di tutti i secondi componenti aventi l'argomento come primo
    dom1: lista di tutti i valori che compaiono come primo componente
    dom2: lista di tutti i valori che compaiono come secondo componente
oltre ad un costruttore che applicato ad una sequenza (anche vuota) di coppie del
correttotipo costruisce una relazione con solo tali coppie, e agli usuali additio-
nals per la presentazione, l'uguaglianza e la duplicazione.
Si forniscano in OCaml e per valori IMMUTABLE:
 (a) API
 (a) Stato concreto e funzioni AF ed I per un ADT;
 (d) Le operazioni elencate per tale ADT.
 (e) Un caso di uso per ogni operazione.

Esercizio (Funzione come specializzazioe di Relazione Binaria in Java)
Si assuma dato un ADT Rel2.java che implementa in Java la relazione binaria polimo
rfa, con le operazioni introdotte sopra, come valore MUTABLE.
Si definisca ora un nuovo ADT, Fun.java, che specializzi Rel2.java per la sotto-
classe delle relazioni che sono funzioni finite. Allo scopo, si forniscano:
 (a) Definizione della APIRel2.java
 (b) intestazione della classe Fun.java
 (c) Stato concreto e funzioni AF ed I per Fun.java
 (d) definizione operazioni e additionals di Rel2.java overridden in Fun.java
 (e) definizione delle operazioni:
        apply(f,v): valore calcolato da f(v)
        dom(f): dominio di definizione di f
 (f) un caso d'uso per una funzione avente dominio con almeno 5 valori.

Esercizio (Funzione, efficiente)
Come in (a)-(f) di esercizio precedente ma FunO.java richiede che il generico A
sia sottoclasse dell'interfaccia Comparable, ovvero i valori di tipo A abbiamo un
ordinamento totale. Utilizzando questo ordinamento il metodo apply usa ricerca
```

binaria per accedere al valore calcolato dalla funzione sull'argomento dato.

Esercizi su uso di higher Order, tail recursion e functional iterators in Ocaml

esercizio.
Fornire in Ocaml, una definizione tail recursive e iterativa della funzione:
    nth: (n:int) -> (l:'a list) -> 'a
che restituisce lo n-esimo elemento di una lista l se n <= length(l). Altrimenti solleva eccezione.

esercizio.
Fornire in Ocaml, una definizione tail recursive e iterativa della funzione:
    init: (n:int) -> (f:int->'a) -> 'a list
che restituisce la lista delle applicazioni di f ad ogni intero nella successione 0..n-1, ovvero [f 0; f 1; ... ; f (n-1)]. Solleva eccezione quando n ...

esercizio
Fornire in Ocaml, una definizione tail recursive e iterativa della funzione:
    map2: (g: 'a->'b'->'c) -> (l1: 'a list) -> (l2: 'b list) -> 'c list
che applicata a 2 liste di stessa lunghezza, restituisce la lista avente come elemento i-esimo l'applicazione di g all'i-esimo di l1 e di l2 rispettivamente. Altrimenti solleva eccezione.

Altri esercizi li ottieni fornendo soluzioni tail recursive e iterative in sostituzione al codice OCaml che hai fornire per la programmazione dei tipi di dato degli esercizi sopra.