Federico Bernacca, 536683
Per questo progetto ho effettuato le seguenti scelte progettuali:
Come prima cosa ho aggiunto in “type exp” il costrutto “Estring of string”, e di conseguenza in “type evT” il costrutto “String of string” per poter esprimere il tipo “string” nell’ambiente.
Inoltre, in “type evT” ho aggiunto il costrutto “Exception of string” che viene usato per sollevare “un’eccezione” se l’operazione non va a buon fine, al posto di un “failwith” che farebbe terminare il programma.
Per implementare il nuovo tipo di dato dizionario ho creato la struttura “Dict” in “type exp”, che diventa esprimibile nell’ambiente con “DictVal” in “type evT”.
Inoltre, ho aggiunto il costrutto “isEquals” che controlla se due dizionari contengono i soliti elementi restituendo di conseguenza un Bool b.
Di seguito la sintassi astratta del dizionario e delle relative operazioni:

```
Type exp = …

// valore esprimibile di tipo string
| Estring of string

 // elemento di chiave “ide” da cercare nel dizionario “exp”
| Read of ide * exp

// elemento di chiave “ide” con valore “exp” da aggiungere nel terzo elemento della tripla dizionario “exp”
| Add of ide * exp * exp

// elemento di chiave “ide” da eliminare dal dizionario “exp”
| Remove of ide * exp

// elimina tutti gli elementi dal dizionario “exp”
| Clear of exp

// applica la funzione denotata dal primo parametro al valore associato a ogni elemento del dizionario denotato dal secondo parametro
| ApplyOver of exp * exp

// controlla se due dizionari contengono gli stessi elementi
| IsEquals of exp * exp

// dichiarazione dizionario
| Dict of dict
  and dict = Empty | Item of ide * exp * dict;;
```

```
Type evT = …

| String of string

| Exception of string

// valore del dizionario
| DictVal of evDict
  and evDict = (ide * evT) list
```

In aggiunta alla sintassi concreta fornita si ha quella per l’operazione isEquals, esempio:

```
isEquals(my_dict, my_dict)
```

```
- : true
```

Scelte implementative:

- in “Item”, ide è una stringa e rappresenta la chiave univoca, exp è il valore associato alla chiave, e il terzo elemento della tripla può essere Empty o a sua volta un altro Item.
- nell’ambiente, il dizionario è rappresentato da una lista di coppie (ide * evT) list.
- l’operazione “Read of ide * exp” restituisce “Exception of string” se la chiave “ide” non è presente nel dizionario, o se si prova a passare come “exp” un tipo non “dict of Dict”, altrimenti restituisce l’oggetto associato alla chiave.
- l’operazione “Add of ide * exp * exp” restituisce “Exception of string” se la chiave “ide” è già nel dizionario, o se si prova a passare come terzo elemento “exp” un tipo non “dict of Dict”, altrimenti aggiunge in testa la coppia “ide * exp”.
- l’operazione “Remove of ide * exp” restituisce “Exception of string” se la chiave “ide” non è presente nel dizionario, o se si prova a passare come “exp” un tipo non “dict of Dict”, altrimenti rimuove l’oggetto associato alla chiave.
- l’operazione “Clear of exp” restituisce “Exception of string” se si prova a passare come “exp” un tipo non “dict of Dict”, altrimenti svuota il dizionario.
- l’operazione “ApplyOver of exp * exp” restituisce “Exception of string” se si prova a passare come primo “exp” un tipo non “Fun of ide * exp”, o come secondo exp un tipo non “dict of Dict”, altrimenti applica funzioni al dizionario che prendono come parametro elementi interi e restituisce un nuovo dizionario.

I testcase si trovano in dizionario.ml insieme a tutta l’implementazione, e testano tutte le operazioni.