Esercizi Corso Fondamenti di Informatica

Marta Cialdea Mayer e Roberto Maieli

22 marzo 2004

1 Esercizi relativi alle slides dei pacchetti 03 e 04

1. Marcate le risposte corrette:

   - In un linguaggio funzionale:
     (a) le funzioni sono sempre definite per tutti i propri argomenti (sono tutte funzioni totali)
     (b) le funzioni possono essere applicate a qualsiasi argomento
     (c) le funzioni possono essere elementi di una coppia
     (d) è possibile controllare se due funzioni sono uguali mediante il predicato di uguaglianza `=`
     (e) le funzioni possono essere argomenti o valori di altre funzioni
   - Una funzione polimorfa:
     (a) è una funzione che può riportare valori diversi quando è applicata allo stesso argomento
     (b) è una funzione definita sempre sia sugli interi che sui reali
     (c) è una funzione che ha più di un tipo
     (d) è una funzione che si può applicare a qualsiasi argomento, di qualsiasi tipo
   - Secondo il paradigma della programmazione funzionale:
     (a) un programma è un insieme di istruzioni che determinano come modificare il contenuto delle celle di memoria
     (b) un programma è una teoria logica ed eseguire il programma equivale a dimostrare una proposizione
     (c) un programma è una funzione ed eseguire un programma equivale a calcolare un valore
     (d) il calcolo consiste nella risoluzione di un problema
     (e) il calcolo consiste nella riduzione di un'espressione ad una più semplice

- Sia F una funzione di tipo:

$$\alpha \times \beta \times \alpha \rightarrow \alpha \times \beta$$

Quali dei seguenti valori possono essere argomenti di F?

valore	SI	NO
`(3, true, 3.0)`		
`(true, true, false)`		
`(10, "pippo", 20)`		
`("pippo", 30, true)`		

2. Definire una funzione Ocaml che, applicata a un intero `n`, riporti `n * 10`.

3. Definire una funzione Ocaml che calcoli il valore assoluto di un numero intero.

4. Definire una funzione Ocaml che calcoli sia il predecessore che il successore di un numero intero.

5. Definire una funzione Ocaml che valuti se un intero è pari o dispari.

6. Scrivere una funzione Ocaml che, applicata a tre interi, ne riporti il minimo. Qual è il suo tipo?

7. Scrivere un programma in Ocaml, che dato il giorno e il mese, `(giorno, mese)`, calcoli il giorno successivo; esempio dato il giorno `(4, marzo)`, restituisce la data `(5, marzo)`, invece, dato il giorno `(31, marzo)` restituisce la data `(1, aprile)`.

8. Dopo aver immesso le seguenti dichiarazioni:

```
let y = 100;;
let x = let y=10 in 3*y;;
```

qual è il valore dell'espressione `y * x`?

9. Dopo aver immesso le seguenti dichiarazioni:

```
let y = "ciao ";;
let x = let y="caro " in y ^ "pippo";;
```

qual è il valore dell'espressione `y ^ x`?

10. Dopo aver immesso le seguenti dichiarazioni:

```
let y = 10;;
let x = let y=20 in 3+y;;
```

qual è il valore dell'espressione `y * x`?

11. Dopo aver immesso le seguenti dichiarazioni:

```
let y = "pluto";;
let x = let y="caro " in y ^ "amico ";;
```

qual è il valore dell'espressione `x ^ y`?

12. Dopo aver immesso le seguenti dichiarazioni:

```
let y = 50;;
let x = let y=100 in y-20;;
```

qual è il valore dell'espressione `y + x`?

13. Dopo aver immesso le seguenti dichiarazioni:

```
let y = "ciao ";;
let x = let y="minnie " in y ^ "cara";;
```

qual è il valore dell'espressione `y ^ x`?

14. Si immettano le seguenti dichiarazioni, nell'ordine in cui sono date:

```
# let x = 10;;
# let y = x * 2;;
# let f(n) = n + y;;
# let x = 100;;
```

Qual è ora il valore di `y`? Quale quello di `f(3)`?

15. Sia `times` la funzione così definita:

```
let rec times (n,m) = if (n=0 or m=0)  then 0 else m + times(n-1,m)
```

Qual è il tipo di `times`? Che cosa calcola `times (n,m)`, quando `n` e `m` sono numeri naturali?

Illustrare i passi della riduzione dell'espressione `times (3,5)` fino ad ottenere un valore.

16. Sia `resto` la funzione così definita:

```
let rec resto (n,m) = if n<m then n else resto (n-m,m)
```

Qual è il tipo di `resto`?

Illustrare i passi della riduzione dell'espressione `resto (10,3)` fino ad ottenere un valore.

Che cosa calcola, in generale, `resto (n,m)`, quando `n` e `m` sono numeri interi positivi?

17. Sia `divide` la funzione così definita

```
let rec divide (n, m) =
    if (n mod m)=0 then m else divide (n, m-1)
```

Qualè il tipo di `divide` ? Illustrare i passi di riduzione dell'espressione `divide (5, 3)` fino ad ottenere un valore. Cosa calcola in generale `divide` quando `m` ed `n` sono numeri interi ?

18. Sia `f` una funzione di tipo `(int -> int) -> int`.
L'espressione `f 3` è corretta? Se lo è, qual è il suo tipo?

Sia `g` la funzione così definita:

```
let rec g n =
   if n=0 then 1
   else n * g (n-1)
```

L'espressione `f g` è corretta? Se lo è, qual è il suo tipo?

19. Sia `g` la funzione definita al punto precedente. Si illustrino i passi della riduzione dell'espressione `g 5` fino ad ottenere un valore.

20. Si consideri la seguente dichiarazione:

```
let rec f(x,y) =
  if x>y then 1
  else x * f(x+2,y)
```

Qual è il tipo di `f`? Qual è il valore di `f(2,6)` ? Che cosa calcola `f`, in generale? (darne una specifica dichiarativa)

21. Si consideri la seguente dichiarazione:

```
let rec g(x,y) =
  if y<x then 1
  else g(x,y-1) * y
```

Qual è il tipo di `g` ? Qual è il valore di `g(3,7)`? Che cosa calcola `g`, in generale? (darne una specifica dichiarativa)

22. Si consideri la seguente dichiarazione:

```
let rec h(x,y) =
  if y<x then 0
  else x + h(x+2,y)
```

Qual è il tipo di `h` ? Qual è il valore di `h(3,10)` ? Che cosa calcola `h`, in generale? (darne una specifica dichiarativa)

23. Si consideri la seguente dichiarazione:

```
let rec k(x,y) =
  if x>y then 0
  else h(x,y-1) + y
```

Qual è il tipo di `k` ? Qual è il valore di `k(4,8)` ? Che cosa calcola `k`, in generale? (darne una specifica dichiarativa).

24. Si consideri la seguente dichiarazione:

```
let rec f (x,y,div) =
  if x>y then 0
  else if x mod div = 0
       then x + f (x+1,y,div)
       else f (x+1,y,div)
```

Qual è il tipo di `f`? Qual è il valore di `f(1,7,2)`? (si noti che `x mod div = 0` se `div` è un divisore di `x`). Che cosa calcola `f`, in generale? (darne una specifica dichiarativa)

25. Si consideri il seguente programma:

```
(* double: int -> int *)
let double n = 2 * n;;

(* sd: int * int -> int *)
let rec sd (min,max) =
  if max < min then 0
  else double min + sd(min+1,max)
```

Si illustrino i passi della riduzione dell'espressione `sd(2,4)` fino ad ottenere un valore.

26. Dato la funzione ocaml `gcd` che calcoli il massimo comun divisore di due interi

```
let rec gcd(m,n) = if m=0 then n else gcd(n mod m,m)
```

ridurre manualmente, fin dove possibile, l'espressione `gcd(40,15)`.

27. Definire una funzione che, applicata a una coppia di numeri interi (n, k), restituisca il primo divisore di k compreso tra 1 e n (inclusi). La funzione solleva un'eccezione se $n \leq 0$.

28. Sfruttando, eventualmente la funzione definita al punto precedente, definire una funzione che data una coppia di interi (n, k) restituisca il primo divisore pari compreso tra 1 ed n (incluso). La funzione solleva un'eccezione se $n \leq 0$.

29. Definire una funzione che determini se un numero è *primo* (quindi divisibile solo per 1 e per se stesso).

30. Assumendo che sia stata definita la funzione `gcd: int * int -> int` che, applicata a una coppia di interi positivi (n, m) riporta il massimo comun divisore di n e m. definire una funzione che, applicata a una coppia di numeri interi (n, k), con k positivo, determini se esiste un *numero relativamente primo* con k compreso tra 1 e n (inclusi). Diremo che m è relativamente primo con k se $gcd(m, k) = 1$. Specificare il tipo della funzione definita.

31. Definire una funzione che applicata ad un intero n positivo restituisca la stringa ottenuta concatenando le stringhe che corrispondono agli interi compresi tra 1 ed n inclusi. (Suggerimento: si usi la funzione Ocaml predefinita `string_of_string` che applicata ad un intero riporta la stringa corrispondente a tale intero; ad esempio `string_of_int 21=''21''`).

32. Definire una funzione tale che dati tre interi (n, m, p) restituisca p meno la somma dei numeri compresi tra n ed m. La funzione solleva un'eccezione se $n > m$ oppure se la somma è $> p$. (Suggerimento: si sfrutti la funzione `sumbetween`, definita a pagina 36 del blocco 04 delle slides).

33. Trasformare le funzioni definite nei punti 27-32 in funzioni definite in maniera puramente locale.