Programmazione Funzionale e Parallela

Corso di Laurea in Ingegneria Informatica e Automatica - A.A. 2019-2020

[T01] Esercitazione PFP del 2 marzo 2020

Istruzioni per l'esercitazione:

- Aprite il form di consegna in un browser e loggatevi con le vostre credenziali `uniroma1`.
- Scaricate e decomprimete sulla scrivania il codice dell'esercitazione. Vi sarà una sotto-directory separata per ciascun esercizio di programmazione. Non modificate in alcun modo i programmi di test `E*Main.scala`.
- Rinominare la directory chiamandola `cognome.nome`. Sulle postazioni del laboratorio sarà `/home/studente/Desktop/cognome.nome/`.
- È possibile consultare appunti/libri e il materiale didattico online.
- Rispondete alle domande online sul modulo di consegna.
- **Finiti gli esercizi**, e non più tardi della fine della lezione:
  - **zippate la directory di lavoro** in `cognome.nome.zip` (`zip -r cognome.nome.zip cognome.nome/`).
- **Per consegnare**:
  - inserite nel form di consegna come autovalutazione il punteggio di ciascuno dei test forniti (inserite zero se l'esercizio non è stato svolto, non compila, o dà errore di esecuzione).
  - fate **upload** del file `cognome.nome.zip`.
  - **importante**: fate logout dal vostro account Google!
- **Prima di uscire**:
  - eliminate dal desktop la directory creata (`rm -rf cognome.nome`).
  - firmate il **foglio presenze**.
  - rimettete a posto eventuali **sedie** prese all'ingresso dell'aula!

Esercizio 1 (somma primi n numeri con ricorsione di coda)

Si modifichi la realizzazione del seguente metodo `sum` che calcola la somma dei primi n numeri in modo che esibisca ricorsione di coda:

```
object E1 {
    def sum(n:Int):Int = if (n<1) 0 else n+sum(n-1)
}
```

Scrivere la soluzione nel file `E1.scala` in modo simile a quanto fatto negli esercizi precedenti e usare il programma di prova `E1Main.scala` fornito.

Si verifichi che la realizzazione è effettivamente con ricorsione di coda usando l'annotazione del compilatore `@scala.annotation.tailrec` (si veda la dispensa).

Per compilare da riga di comando usare: `scalac E1Main.scala E1.scala`. Si noti che sulla riga di comando ci sono entrambi i file che compongono il programma. Noterete la presenza di vari file `.class` generati dalla compilazione. Non serve eliminarli prima di consegnare il compito.

Per eseguire il programma da riga di comando usare: `scala E1Main`. Si noti che, come in Java, al comando `scala` viene passato il nome della classe (come vedremo, `object` denota una classe in cui tutti i metodi/variabili sono statici) da cui parte il programma.

Esercizio 2 (massimo comun divisore)

Si scriva un metodo Scala `mcd` che calcola il massimo comun divisore (MCD) di due numeri interi. Usare la seguente definizione ricorsiva:

```
MCD(x,y)=x, se y=0
MCD(x,y)=MCD(y, resto della divisione di x per y) altrimenti
```

Compilare ed eseguire come visto nell'Esercizio 1. Scrivere la soluzione nel file `E2.scala` in modo simile a quanto fatto negli esercizi precedenti e usare il programma di prova `E2Main.scala` fornito.

Esercizio 3 (somma dei quadrati)

Si scriva un metodo Scala `sommaQuadrati` che, dati due interi `x` e `y` con `x<=y`, calcola la somma dei quadrati dei numeri da `x` a `y`, compresi. Scrivere la soluzione nel file `E3.scala` in modo simile a quanto fatto negli esercizi precedenti e usare il programma di prova `E3Main.scala` fornito.

Esercizio 4 (uguaglianza parziale di funzioni)

Scrivere una funzione Scala che, date due funzioni `f1:Int=>Int` e `f2:Int=>Int` e un intero `n`, verifica che `f1` e `f2` calcolino lo stesso valore su ogni input compreso tra `0` e `n` (inclusi). La funzione deve restituire un `Boolean`. Scrivere la soluzione nel file `E4.scala` in modo simile a quanto fatto negli esercizi precedenti e usare il programma di prova `E4Main.scala` fornito.

Esercizio 5 (somma di f(x))

Scrivere un metodo Scala `somma` che, dato come parametro una funzione `f:Int=>Int`, restituisce una funzione che prende come parametri due interi `a` e `b` e restituisce la somma di `f(x)` per ogni `x` compreso tra `a` e `b` (estremi inclusi). Scrivere la soluzione nel file `E5.scala` in modo simile a quanto fatto negli esercizi precedenti e usare il programma di prova `E5Main.scala` fornito.

Suggerimento. Usare il seguente schema nella soluzione:

```
object E5 {
    def somma(...) = {
        def funzioneDaRestituire(...) ... = ... // metodo definito localmente (ispirarsi all'esercizio 3)
        funzioneDaRestituire _              // restituisce metodo locale convertito a funzione con _
    }
}
```