Programmazione Funzionale e Parallela

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

Esercitazione del 6 aprile 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 oltre le 23:59 :
  - **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`.
- È possibile consultare la documentazione delle API di Scala, in particolare quelle sulle liste, e la dispensa Scala.
- **Se avete domande** accedete a Google Meet all'indirizzo meet.google.com/sph-eiax-fsv durante orario 14:00-16:00 stabilito per l'esercitazione accedendo con la vostra **mail istituzionale**. Troverete online il docente e il tutor del corso. In alternativa, scrivete via email.

Per maggiori informazioni fate riferimento al regolamento delle esercitazioni.

Esercizio 1 (prodotto scalare)

Scrivere un metodo `scalarProd` che, date due vettori rappresentati come sequenze di `Double`, ne calcola il prodotto scalare. Se i vettori hanno lunghezze diverse, limitare il prodotto scalare al range di indici validi comuni. Ad esempio: `scalarProd(Seq(3,4), Seq(2,9,1)) == 3*2 + 4*9 == 42`.

Suggerimento: usare ricorsione oppure i metodi delle collezioni (potrebbero essercene di utili anche fra quelli non visti a lezione).

Usare il main di prova incluso nel codice dell'esercitazione scaricato.

Esercizio 2 (metodi impliciti)

Scrivere un metodo `isMappedFrom`, applicabile a un `Vector v` che verifica se un altro `Vector m` è ottenibile da `v` applicando la funzione `f` a ciascun elemento di `v`.

Usare il main di prova incluso nel codice dell'esercitazione scaricato.

Esercizio 3 (un algoritmo di sorting da premio Ignobel)

Se esistesse un premio Ignobel per l'informatica, questo si piazzerebbe bene. Scrivere un metodo `noobSort` che, dato un `Vector v` di `n` elementi di tipo generico, restituisce la versione ordinata di `v`. Per risolvere il problema, generare tutte le permutazioni degli indici da `0` a `n-1` e per ciascuna permutazione generare il vettore permutato, verificare se è ordinato ed eventualmente restituirlo.

Suggerimento: per generare le permutazioni degli indici, usare `(0 to n-1).permutations`. In alternativa si può permutare direttamente il vettore di input e trovare quella permutazione per cui è ordinato. Si consulti la documentazione delle API Scala per cercare metodi utili.

Usare il main di prova incluso nel codice dell'esercitazione scaricato.

Esercizio 4 (passaggio per nome)

Si vuole aggiungere un costrutto Scala `repeat` che, dato un intero `n` e un corpo `body`, esegue `body` per `n` volte come nel seguente esempio:

```
repeat(5) {
    println("test")
}
```