Programmazione Funzionale e Parallela (A.A. 2015-2016)
Corso di Laurea in Ingegneria Informatica e Automatica
Sapienza Università di Roma

C

Esame del 14/01/2016 – Durata 1h 30' (non esonerati)

Inserire nome, cognome e matricola nel file `studente.txt`.

Esercizio 1

Completare il file `C1.scala` in modo che sia possibile compilare ed eseguire correttamente il seguente programma di prova `C1Main.scala`:

```
import Point._

object C1Main extends App {
    val c  = Circle(1,1,5)
    val l1 = List(Point(1,1), Point(5,7), Point(1,2), Point(1,0))
    val l2 = List(Point(1,1), Point(1,2))

    val n1:Boolean     = c >? l1  // c contiene tutti i punti in l1?
    val n2:Boolean     = c >? l2  // c contiene tutti i punti in l2?
    val p1:List[Point] = c >> l1  // punti di l1 contenuti in c
    val p2:List[Point] = c >> l2  // punti di l2 contenuti in c
    val s:String       = "punto=" + Point(9,4) // converte punto a stringa
    val p:List[Point]  = l2 +++ List(Point(13,13)) // concat. liste di punti

    println(s  + " [corretto: punto=(9.0, 4.0)]")
    println(n1 + " [corretto: false]")
    println(p1 + " [corretto: List((1.0,1.0), (1.0,2.0))]")
    println(n2 + " [corretto: true]")
    println(p2 + " [corretto: List((1.0,1.0), (1.0,2.0))]")
    println(p +  " [corretto: List((1.0,1.0), (1.0,2.0)), ((13.0,13.0))]")
}
```

La soluzione non deve usare alcun costrutto della programmazione imperativa e in particolare alcuna variabile `var`.

Esercizio 2

Si vuole scrivere un metodo ricorsivo che calcola il prodotto scalare di vettori rappresentati come liste di `Int`. Scrivere la soluzione nel file `C2.scala` in modo che sia possibile compilare ed eseguire correttamente il seguente programma di prova `C2Main.scala`:

```
object C2Main extends App {
    val l1:List[Int] = List(1,0,1,1)
    val l2:List[Int] = List(1,0,0,1)
    val l3:List[Int] = List(1,1,1,1,4)
    val l4:List[Int] = Nil

    val r1:Int = C2.ps(l1,l2)
    println(r1  + " [corretto: 2]")

    val r2:Int = C2.ps(l1,l3)
    println(r2  + " [corretto: 3]")

    val r3:Int = C2.ps(l1,l4)
    println(r3  + " [corretto: 0]")
}
```

La soluzione non deve usare alcun costrutto della programmazione imperativa e in particolare alcuna variabile `var`.

Esercizio 3 (Ricerca delle occorrenze di una stringa in un testo mediante pthread)

Lo scopo dell'esercizio è quello di scrivere una funzione C che cerca tutte le occorrenze di una stringa in un testo in parallelo usando pthread. Si vada nella directory di lavoro `findst` e si definisca nel file `findst.c` la funzione `finds` con il seguente prototipo:

```
void findst(const char* pattern, const char* text, char* out, int n)
```

dove:

- `pattern`: stringa da cercare;
- `text`: array di `n` caratteri (non terminato con '\0') che rappresenta il testo in cui cercare la stringa;
- `out`: vettore di output di dimensione `n` (si veda sotto come calcolarlo);
- `n`: numero di caratteri del testo.

La funzione deve inizializzare `out` in modo che, per ogni `i` in [0, `n`-1], `out[i]` vale 1 se la stringa `pattern` compare in `text` a partire dall'indice `i`, e 0 altrimenti.

Per compilare usare il comando `make`. Per effettuare dei test usare `make test1` e `make test2`.