Programmazione ad Oggetti
Mod. 2

10/1/2024

Studente _____ Matricola _____

1. Si consideri un metodo statico in Java 8+ di nome `parallelFactorial()` che data una `Collection<Integer>` produce una `Collection<FactorialThread>`. Per ogni intero della collection di input viene effettuato lo *spawning* di un nuovo thread che ne computa il fattoriale e ne conserva il risultato in qualche modo. Tutti i thread creati vengono ritornati nella collection di output, che naturalmente ha la stessa lunghezza della collection di input.
   (a) [4 punti] Si implementi la classe `FactorialThread` in modo che estenda `java.lang.Thread` e fornisca un metodo getter per l'accesso al risultato della computazione del fattoriale. Si presti attenzione all'attesa della fine della computazione, gestendo opportunamente la cosa.
   (b) [4 punti] Si implementi il metodo statico `parallelFactorial()` avente la seguente firma:
   ```
   static Collection<FactorialThread> parallelFactorial(Collection<Integer> c)
   ```
   Ogni thread deve lavorare concorrentemente agli altri, ciascuno calcolando il fattoriale di un intero proveniente dalla collection di input. La funzione `parallelFactorial()` non deve attendere la fine delle computazioni, ma deve ritornare subito la collection di output.
   (c) [1 punti (bonus)] Si raffini la firma del metodo statico `parallelFactorial()` di cui al punto precedente in modo che il tipo di input ed il tipo di output siano, rispettivamente, più generale e più specializzato possibile, senza tuttavia rivelare dettagli sull'implementazione.
   (d) [2 punti] Si scriva uno snippet di codice che testi il metodo statico `parallelFactorial()` stampando i risultati di ciascuna computazione.
   (e) Si dia una seconda implementazione del metodo statico `parallelFactorial()` usando la funzione di ordine superiore `map()`[1]. In particolare si proceda nel seguente modo:
      i. [3 punti] Si implementi un metodo statico `map()` avente la seguente firma:
      `static <A, B> List<B> map(Iterable<A> i, Function<A, B> f)`
      Esso deve applicare la funzione `f` ad ogni elemento di `i` e produrre in output tutti i risultati delle applicazioni.
      ii. [3 punti] Si reimplementi `parallelFactorial()` tramite una singola invocazione della `map()`.

2. Si implementino in C++11 alcune varianti della funzione `map()` rispettando la sua semantica tipica: la funzione passata come argomento alla `map()` deve essere applicata ad ogni elemento della sequenza di input, producendo una sequenza di output con tutti i risultati delle applicazioni.
   (a) [7 punti] Si implementi la seguente versione della `map()` che opera su iteratori. Si badi che in questo caso la funzione non ha tipo di ritorno. L'output consiste in un iteratore passato come argomento: lì la funzione dovrà scrivere i risultati.
   ```
   template <class InputIterator, class OutputIterator>
   void map(InputIterator from, InputIterator to, OutputIterator out,
            function<typename OutputIterator::value_type(typename InputIterator::value_type)> f)
   ```

[1] La funzione `map()` qui richiesta è equivalente a quella che JDK e STL chiamano `transform()`.

(b) [5 punti] Si implementi la seguente versione della `map()` che opera su `vector` di STL. In questo caso l'output è un vero e proprio tipo di ritorno. L'implementazione deve invocare la `map()` di cui al punto precedente.

```
template <class A, class B>
vector<B> map(const vector<A>& v, function<B(A)> f)
```

(c) Il tipo templatizzato `function` è definito da STL a partire dallo standard C++11, così come le espressioni lambda. Nessuno dei due esisteva in C++ *vanilla* (oggi chiamato C++03).

i. [2 punti] Come sarebbe stato possibile scrivere in C++03 le firme delle `map()` di cui ai punti (a) e (b) senza usare il tipo `function`?

ii. [1 punti (bonus)] Le firme compatibili con C++03 potrebbero coesistere con le firme originali di cui ai punti (a) e (b)? In altre parole, sarebbero considerati tutti overload validi dal compilatore C++11?

iii. [2 punti] Come sarebbe stato possibile invocare le `map()` compatibili con C++03 senza le lambda?

Question:	1	2	Total
Points:	16	16	32
Bonus Points:	1	1	2
Score:			