Parte pratica con eclipse – 12 febbraio 2016

Scegli come workspace il desktop o la tua home.

Create un progetto per ognuno degli esercizi con nome “COGNOME_NOME_N” con N = 1, 2, 3, 4

Per la consegna, esportate un file zip per progetto utilizzando il wizard di eclipse (selezionando il progetto che vi interessa) e salvando il tutto in un file zip con nome COGNOME_NOME_N.zip

Caricate tutti gli zip (uno alla volta) nell'unica domanda del quiz. Potete anche sostituire uno zip successivamente.

Puoi caricarlo quando vuoi, anche ricaricarlo successivamente ma assicurati di avere solo un solo zip per progetto. Se ti è utile, includi `<stdbool.h>`

Per passare da una finestra d un'altra usa ALT+TAB

1. Ricorsione - C - divisibile

Scrivi una funzione **divisibile** in C che prende due interi n e i e restituisce true se n è divisibile per qualche numero tra 2 e i.

Al solito, scrivi tre versioni: una non ricorsiva, una ricorsiva senza tail recursion e una ricorsiva con tail recursion. Specifica esattamente i parametri che passi alla procedura, il tipo di passaggio utilizzato e il loro significato.

Scrivi poi una funzione primo che prende un n e restituisce true se il numero è primo utilizzando una delle implementazioni di divisibile che hai fatto.

Scrivi un main di esempio in cui controlli che 17 sia un numero primo. Disegna il record di attivazione (solo per la funzione scelta in primo) fino alla massima estensione del record di attivazione. Nel caso di tail recursion, spiega nel codice quali ottimizzazioni hai adottato o potresti adottare.

Se riesci puoi anche ottimizzare in modo di non provare tutti i numeri da 2 a n per vedere se n è divisibile.

2. Tipi opachi per array di booleani (bitvector)

Per manipolare dei bitvector in modo sicuro si possono usare i tipi opachi.

Definisci un tipo opaco `bitvector` che ha come campi la dimensione e il contenuto vero e proprio come array di qualche tipo primitivo.

I metodi che la libreria mette a disposizione sono:

- creazione di una bitvector (make) di dimensione n inizialmente tutto falso.
- le operazioni logicche AND, OR, NOT. Le operazioni binarie sono fatte a bit a bit solo se i due bitvector hanno dimensione uguale.
- toString che restituisce una string di 0 e 1.
- e un delete che fa da distruttore

Fai un esempio ben commentato in cui utilizzi tutti i metodi sopra.

Ottimizzazione: un char o int in verità può ospitare tanti bit, quindi è inutile usare un intero per ogni bit. Ad esempio se int ha sizeof uguale a 4 byte, sono quindi 32 bit, potrei ospitare 32 bit.

3. Dangling pointer

Fai un esempio in cui hai un dangling pointer e non ti accorgi per un po' e poi ti accorgi che il puntatore non è piu' valido. Commenta il codice.

4. Generics in Java

Una classe astratta generica Function ha due parametri T (ordinabile) e S e definisce un metodo astratto compute che prende un S e calcola il suo T.

Scrivi, ad esempio StringLength è una Function che come compute restituisce la lunghezza di una String.

La class Applier ha un metodo apply che prende una lista di elementi di tipo S e una Function<T,S> f e ordina la lista utilizzando il valore che compute di f restituisce sugli elementi della lista.

Scrivi un esempio in cui ordini una lista di stringhe in base alla loro dimensione usando Applier e StringLength.

Definisci Persona con nome che è comparable a seconda del nome e Studente che estende persona. Se volessi ordinare una lista di studenti in base al loro nome, come posso riusare Applier e Function?

Cerca di usare tutti i costrutti visti come i wildcard, i bounded generics e così via.

5. definizione di funzione in SCALA

Scrivi una funzione **countC7** che dati due interi a e b restituisce il numero di interi tra a e b (compresi) che contengono la cifra 7.

Per ottenere una lista tra due interi a e b puoi usare **List.range(a, b)**

Definisci e utilizza una funzione **contieni7** che controlla se un numero contiene la cifra 7.

Fai diverse versioni (in ordine di difficoltà)

- **countC7** con un semplice ciclo for
- **countC7** usando il foreach (e una funzione + var locale)
- **countC7** in modo ricorsivo (usa isEmpty, head e tail)
- **countC7** usando anche map in modo che converta ogni string a 1 se contiene 7, 0 altrimenti
- **countC7** usando foldLeft
- **countC7** usando anche filter

COMMENTA il codice

Se volessi generalizzare il tutto in due modi:

- una funzione contieniN che controlla se un numero contiene n e poi passare questa funzione ad una HOF come dovrei fare? Riscrivi un paio di metodi di cui sopra usando le higher order function e contieniN. Usa il currying se riesci.

Scrivi un po' di chiamate nel main di prova. Scrivi tutto in un

```
object Esercizio {
   def contieni7 ….
   def countC7 ….
   def main(args: Array[String]) {
     ….
   } }
```