Parte pratica con eclipse – 15 gennaio 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.

Puoi caricarlo quando vuoi, anche ricaricarlo successivamente ma assicurati di avere solo un solo zip per progetto.

1. Ricorsione - C - palindorma

Si scriva una funzione **palindorma** in C che restituisce true se la stringa passata come argomento è palindroma
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 anche un main di esempio in cui chiami la funzione con la stringa “anna” e assegna il risultato ad una variabile globale RESULT. Prova a chiamare tutte e tre le versioni della funzione.

Disegna il record di attivazione per tutte e tre le versioni fino alla massima estensione del record di attivazione. Nel caso di tail recursion, spiega nel codice quali ottimizzazioni hai adottato o potresti adottare.

2. Tipi opachi per String

Per manipolare le stringhe in modo sicuro si possono usare il puntatori opachi (come proposto dal CERT). Definisci un tipo opaco `string_mx` che ha come campi la dimensione attuale (size), la dimensione massima (maxsize), e il contenuto vero e proprio (cstr).

I metodi che la libreria mette a disposizione sono:

- creazione di una stringa (make) che prende un array di char zero terminated e crea la string_mx
- strccpy_mx che copia in una string_mx s1 la string_mx s2 (in modo sicuro, se s2 non sta in s1, la tronca)
- tostring che converte una string_mx in una string normale (char *) zero terminated
- e un delete_mx che fa da distruttore

Fai un esempio ben commentato in cui utilizzi tutti i metodi sopra. Riesci a fare un buffer overflow con `string_mx` ?

3. Passaggio per riferimento in C++

Scrivi una funzione **MCD** che prende due interi a e b per riferimento e calcola il quoziente che viene messo in a e il resto che viene messo in b (quindi la funzione è void). Scrivi un main in cui fai alcune prove. Commenta il codice. C'è qualche caso critico in cui la funzione potrebbe non andare bene?

4. Generics in Java

Una dizionario è una lista di entries, ognuna delle quali è una coppia chiave e valore. La chiave però deve essere ordinabile. Definisci la classe Dizionario usando i generics (e una List).

Dizionario ha i seguenti metodi:

- **cerca**(key) → cerca nel dizionario il valore con chiave key e lo restituisce, null se non c'è alcun valore con quella chiave

- **cercaVal**(value) → cerca nel dizionario il valore e restituisce true se presente

- **insert**(key,value) → inserisce la coppia key e value se non è già presente

- **stampa** → stampa il dizionario in ordine delle key

Esiste anche una classe di appoggio che ha un metodo statico

sumkey(dizionario) che dato un dizionario restituisce la somma delle chiavi (assume quindi che le chiavi siano Integer)

Definisci alcuni dizionari (con chiavi e valori di tipo diverso), inserisci alcuni valori, stampali e chiama sumkey (se possibile)

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 **pls** che data una lista di String (List[String]) restituisce il prodotto delle lunghezze delle stringhe contente in essa che iniziano con “a”

Fai diverse versioni (in ordine di difficoltà)

- **pls1** con un semplice ciclo for
- **pls2** usando il foreach (e una funzione + var locale)
- **pls3** in modo ricorsivo (usa isEmpty, head e tail)
- *pls***4** usando folderLeft
- **pls5** usando anche filter (più un metodo dei precedenti)
- **pls6** usando anche map

COMMENTA il codice

Se volessi generalizzare il tutto e calcolare il prodotto dell'applicazione di una funzione **stp** che data una String mi restituisce un Int, come dovrei fare? Riscrivi un paio di metodi di cui sopra usando le higher order function e una funzione stp opportuna . Usa il currying se riesci.

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

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