PROGRAMMAZIONE II – A.A. 2014-15 – Progetto Sessione Autunnale v.2 (15 luglio)

Il progetto ha l'obiettivo di applicare a casi specifici i concetti e le tecniche di programmazione esaminate durante il corso, consiste nella progettazione e realizzazione di alcuni moduli software ed è strutturato in due esercizi di programmazione.

Esercizio 1: Progettazione e sviluppo di un interprete in OCaml

Si consideri un semplice linguaggio funzionale che oltre a operazioni standard su interi e booleani comprende la definizione di un tipo di dati strutturato **tuple**, con relative operazioni. Una tupla è una struttura dati immutabile che contiene una sequenze ordinata di elementi di tipo eterogeneo. Ad esempio, la tupla **t = [1, "hello", "world", true]** contiene quattro elementi: il primo elemento è il valore intero **1** , il secondo elemento è la stringa **"hello"**, etc. Ogni elemento di una tupla è identificato in modo univoco dalla sua posizione e pertanto si può accedere agli elementi di una tupla singolarmente indicandone la posizione. Ad esempio, **t at 2** permette di selezionare l'elemento in seconda posizione della tupla **t**, ovvero il valore **"hello"**. La sintassi concreta del linguaggio è definita dalla grammatica riportata in seguito.

(Identificatori)	Ide ::= <definizione standard>	
(Interi)	Int ::= <definizione standard>	
(Booleani)	Bool ::= <definizione standard>	
(Espressioni)	E ::= Ide | Int | Bool	
	| E and E | E or E | not E	(Espressioni Booleane)
	| OP(E,E)	(Espressioni su interi con OP ∈{"+", "-", "*", "=", "<="})
	| Ide -> E	(Funzione con un parametro, non ricorsiva)
	| if E then E else E	(Condizionale)
	| let Ide = E in E	(Blocco let)
	| Ide(E)	(Applicazione Funzionale)
	| tuple[Elts]	(Espressione tupla)
	| E at Int	(Accesso elemento tupla)
	| E fst Int	(Selezione elementi tupla)
	| E == E	(Confronto tra tuple)
	| Ide@E	(Applica una funzione agli elementi di una tupla)
	Elts ::= E | E ; Elts	(Elementi di una tupla)

1. Si definisca una sintassi astratta per il linguaggio, introducendo opportuni tipi di dati OCaml. Si noti che le tuple possono essere annidate.
2. Si definisca in OCaml un interprete del linguaggio che rispetti le specifiche seguenti
    a. L'applicazione funzionale deve essere valutata con la regola di scoping statico.
    b. La valutazione di un'espressione tupla restituisce una tupla nella quale tutti gli elementi sono valutati. Per esempio, supponendo che l'ambiente corrente contenga il legame tra la variabile **x** e il valore **5**, la valutazione di **tuple[x; x*2]** deve restituire **tuple[5; 10]**.
    c. L'operazione di selezione **exp at i** restituisce il valore dell'**i**-esimo elemento della tupla **exp**. Non è definita se **exp** non è una tupla oppure se non ha elementi alla posizione **i**.

d. L'espressione **exp1 == exp2** è definita solo se entrambi gli argomenti sono tuple, e vale **true** se **exp1** e **exp2** hanno elementi con lo stesso valore nel medesimo ordine.
e. L'espressione **exp fst k** è definita solo se **exp** è una tupla con almeno k elementi (k è un valore intero), e restituisce la tupla composta dai primi **k** elementi della tupla **exp**.
f. L'applicazione **ide@exp** è definita solamente se **ide** è una funzione ed **exp** è una tupla di valori omogenei. In questo caso il risultato è la tupla ottenuta applicando la funzione **ide** a tutti gli elementi della tupla passata come argomento.
g. Tutte le altre operazioni hanno il significato visto a lezione. Per quanto non specificato, si documentino e motivino le scelte fatte nella relazione.

3. Si verifichi la correttezza dell'interprete progettando ed eseguendo una quantità di casi di test sufficiente a testare tutti gli operatori.
4. Si traduca nella sintassi astratta proposta la seguente espressione, e la si valuti con l'interprete.

```
let add5 x =  x + 5 in
   let t = tuple[5, 6, true, 7] in
      add5@(t fst 2)
```

Il valore atteso è **`tuple[10, 11]`.**

Esercizio 2: Progettazione e realizzazione di un modulo Java

Si realizzi un modulo Java che simuli la gestione di un sistema di condivisione di documenti digitali. L'applicazione Java deve fornire le funzionalità per la gestione della *repository* dei documenti. Gli utenti devono poter prendere visione di, copiare, inserire ed eliminare documenti digitali. Ogni utente ha un nome (unico nel sistema) e una password.

Il programma deve essere costituito da un numero adeguato di classi per modellare le varie entità coinvolte: in particolare, ci dovrà essere una classe chiamata **DigitalDoc** con l'ovvio significato. Le strutture dati e le funzionalità devono essere distribuite tra le varie classi in modo da garantire l'incapsulamento.

Il sistema deve realizzare funzionalità per due tipologie di utenti: l'operatore e i clienti. L'operatore può creare un nuovo utente (assicurando l'unicità del nome e della password) o eliminarne uno già esistente (e di conseguenza tutti i suoi documenti digitali). Gli utenti (che sono i proprietari dei documenti digitali) possono inserire, leggere ed eliminare i propri documenti digitali o condividere un documento digitale con un altro utente. Le varie operazioni possono lanciare opportune eccezioni, i cui nomi sono indicati nell'interfaccia del punto seguente.

1. Per testare le funzionalità realizzate nel progetto (incluse le eccezioni che esse possono lanciare), si realizzi una classe **ShareDocImpl** che implementi l'interfaccia **ShareDoc** riportata di seguito.

```
public interface ShareDoc {

   // Aggiunge l'utente con la relativa password alla repository.
   // Restituisce true se l'inserimento ha successo,
   // false se fallisce perche' esiste gia' un utente con il medesimo nome.
   public boolean addUser(String name, int password);

   // Elimina l'utente e tutti i suoi documenti digitali
   public void removeUser(String name);

   // Aggiunge al sistema il documento digitale identificato dal nome.
   // Restituisce true se l'inserimento ha successo,
   // false se fallisce perche' esiste gia' un documento con quel nome.
   public boolean addDoc(String user, String doc, int password);
```

```
    // Rimuove dal sistema il documento digitale identificato dal nome.
    // Restituisce true se l'operazione ha successo,
    // false se fallisce perche' non esiste un documento con quel nome.
    public boolean removeDoc(String user, String doc, int password);

    // Legge il documento digitale identificato dal nome.
    // Lancia WrongIdException se user non e' il nome di un utente registrato
    // o se non esiste un documento con quel nome
    // o se la password non e' corretta
    public void readDoc(String user, String doc, int password);

    // Notifica una condivisione di documento
    // Lancia un'eccezione WrongIdException se fromName o toName non sono nomi
    // di utenti registrati o se non esiste un documento con quel nome
    // o se la password non e' corretta
    public void shareDoc(String fromName, String toName, String doc,
                         int password) throw WrongIdException;

    // Restituisce il nome del documento condiviso cancellandolo dalla coda
    // delle notifiche di condivisione.
    // Lancia un'eccezione EmptyQueueException se non ci sono notifiche,
    // o WrongIdException se user non e' il nome di un utente registrato
    // o se la password non e' corretta
    public String getNext(String user, int password) throw EmptQueueException,
                          WrongIdException;
}
```

2. Si definisca una specifica completa comprendente l'*overview* e l'invariante di rappresentazione, e per ogni metodo si fornisca la dimostrazione che l'invariante è preservato.
3. Si realizzi una batteria di test per verificare la correttezza del progetto realizzato.

Modalità di consegna Progetto Sessione Autunnale

- Il progetto deve essere svolto e discusso col docente individualmente. Il confronto con colleghi mirante a valutare soluzioni alternative durante la fase di progetto è incoraggiato,
- Il progetto vale per l'appello di settembre e per l'appello straordinario. Per poter sostenere l'orale dell'appello di settembre 2015, il progetto deve essere consegnato entro le ore 24 di Domenica 6 settembre.
- Il progetto deve essere costituito da:
    - I file sorgente contenenti il codice sviluppato e le corrispondenti batterie di test; tutto il codice deve essere adeguatamente commentato nei file sorgente.
    - Una relazione di massimo una pagina per esercizio, che descrive le principali scelte progettuali ed eventuali istruzioni per eseguire il codice.
- La consegna va fatta inviando per email tutti i file in un archivio. Per il corso A, inviare l'email al Prof. Ferrari con oggetto "[PR2A] Consegna progetto". Per il corso B, inviare l'email al Prof. Corradini con oggetto contenente la stringa "[PR2B] Consegna progetto".

Altre informazioni

- Per quanto riguarda il progetto, i docenti risponderanno solo a eventuali domande riguardanti l'interpretazione del testo, e non valuteranno o commenteranno soluzioni parziali prima della consegna.