# PROGRAMMAZIONE II – A.A. 2016-17 – Secondo Progetto (15 dicembre) v.1

Il progetto ha l'obiettivo di applicare a casi specifici i concetti e le tecniche di programmazione esaminate durante la seconda parte del corso, e consiste nella progettazione e realizzazione di alcuni moduli software.

**Descrizione: Progettazione e sviluppo di un interprete in OCaml**

Si consideri un'estensione del linguaggio didattico funzionale presentato che manipoli tuple di espressioni, e che permetta di combinare opportunamente funzioni: ad esempio è possibile invocare tuple di funzioni. L'estensione minimale dei tipi è riportata di seguito

```
type exp = ... | Etup of tuple          (* le tuple sono anche espressioni *)
        | Pipe of tuple                 (* concatenazione di funzioni *)
        | ManyTimes of int * exp        (* esecuzione iterata di funzione *)
and tuple = Nil                         (* tupla vuota *)
        | Seq of exp * tuple            (* tupla di espressioni *)
```

Gli operatori per il tipo *tuple* hanno il significato intuitivo descritto tra parentesi, ma il secondo e il terzo per *exp* (Pipe e MoreTimes) meritano un commento specifico.

Una nota applicazione dell'idea di pipe si ha nella programmazione di shell, e consiste nel considerare una tupla comprendente solo funzioni unarie (tranne la costante *Nil*) e restituire la funzione ottenuta dalla applicazione in sequenze di dette funzioni. Ad esempio, assumendo di avere *m* funzioni unarie *f1 … fm*, la valutazione dell'espressione Pipe (Seq (f1, Seq (f2 … Seq (fm, Nil) …))) applicata a una espressione *e* coincide intuitivamente con la valutazione dell'espressione Apply (fm, … Apply(f2, Apply(f1, e))…).

Più semplicemente, ManyTimes(m, f) è la funzione ottenuta applicando *m* volte la funzione *f*.

1. Si estenda l'interprete OCaml del linguaggio funzionale assumendo la regola di scoping statico.
2. Si fornisca di conseguenza il type checker dinamico del linguaggio risultante.
3. Si verifichi la correttezza dell'interprete progettando ed eseguendo una quantità di casi di test sufficiente a testare tutti gli operatori aggiuntivi.

La sintassi astratta suggerita può essere modificata e, se ritenuto necessario, estesa.

**Modalità di consegna**

- 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 deve essere costituito da
    - i file sorgente contenenti il codice sviluppato e le corrispondenti batterie di test, ove tutto il codice deve essere adeguatamente commentato;
    - una relazione di massimo una pagina 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 entro il 15 Gennaio 2017. Per il corso A, inviare l'email al Prof. Ferrari con oggetto "[PR2A] Consegna progetto 2". Per il corso B, inviare l'email al Prof. Gadducci con oggetto contenente la stringa "[PR2B] Consegna progetto 2".

**Altre informazioni**

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