# Paradigmi di Programmazione - A.A. 2021-22

Esempio di Testo d'Esame n. 1

**CRITERI DI VALUTAZIONE:**
La prova è superata se si ottengono almeno 12 punti negli esercizi 1,2,3 e almeno 18 punti complessivamente.

## Esercizio 1 [Punti 4]

Applicare la $\beta$-riduzione alla seguente $\lambda$-espressione fino a raggiungere una espressione non ulteriormente riducibile o ad accorgersi che la derivazione è infinita:

$$(\lambda x.xxy)(\lambda x.\lambda y.xyy)$$

Nella soluzione, mostrare tutti i passi di riduzione calcolati sottolineando ad ogni passo la porzione di espressione a cui si applica la $\beta$-riduzione (redex) ed evidenziando le eventuali $\alpha$-conversioni.

## Esercizio 2 [Punti 4]

Indicare il tipo della seguente funzione OCaml, mostrando i passi fatti per inferirlo:

```
let f x y z =
  match x with
  | [ ] -> z
  | w::ws -> w y;;
```

## Esercizio 3 [Punti 7]

Definire, usando i costrutti di programmazione funzionale in OCaml, una funzione `duemax` con tipo

```
duemax : 'a list -> ('a * 'a) option
```

in modo che `duemax lis` restituisca `Some (x,y)` tale che per ogni elemento `z` di `lis` vale `x` $\geq$ `y` $\geq$ `z`. La funzione restituisce `None` in caso di lista vuota. Esempi:

```
duemax [2;3;1;5] = Some (5,3)
duemax ['t'] = Some ('t','t')
duemax [] = None
```

## Esercizio 4 [Punti 15]

Estendere il linguaggio MiniCaml visto a lezione con una nuova forma di astrazione funzionale `both-fun`. L'astrazione consente di definire una coppia di funzioni non ricorsive. In sintassi concreta l'astrazione `both-fun(fun x=x+1, fun x=x*2)` applicata al valore `5` produce come risultato la coppia `<6,10>`.
Definire le regole di valutazione di `both-fun` e estendere consistentemente l'implementazione in OCaml dell'interprete del linguaggio.