# Esame di Programmazione (mod A) - CdL AIDA

Appello I - Gennaio 2022

Giulio Caravagna (gcaravagna@units.it)

L'appello contiene **6** esercizi (`A1`, `A2`, `A3`, `B1`, `B2`, `B3`) da risolvere in 3 ore senza l'ausilio del computer.

**Importante.** `A1`, `A2` e `A3` sono di *sbarramento* e permettono di raggiungere 18/30. `B1`, `B2` e `B3` valgono fino al raggiungimento del voto massimo di 30/30.

## 1 Esercizi di sbarramento (18 punti)

**A1.** Si scrivano due funzioni C: `myfun_iter`, *iterativa*, e `myfun_rec`, *ricorsiva*, che calcolino, per un dato $n \geq 1$ in input, la successione:

$$\begin{cases} a_1 = -1 \\ a_2 = 0 \\ a_n = (\frac{1}{2}a_{n-2} - a_{n-1} + 1) & \text{con } n \geq 3 \text{ se } n \text{ è pari} \\ a_n = (2n + a_{n-1} + 1 + 2a_{n-2}) & \text{con } n \geq 3 \text{ se } n \text{ è dispari} \end{cases}$$

**A2.** Si scriva una funzione *ricorsiva* C che prenda in input due array di interi $v$ e $c$ di uguale dimensione $n > 0$, dove:

- $v$ è un vettore contenente i voti, da 0 a 30, di uno studente;
- $c$ è un vettore contenente il numero di crediti degli esami registrati nel vettore $v$ (quindi l'esame con voto $v_i$ vale $c_i$ crediti, per $0 \leq i < n$);

e calcoli la media *pesata*

$$\mu_{cv} = \sum_{i=0}^{n-1} I(v_i, c_i)$$

ottenuta solamente sugli esami il cui voto sia $> 18$, quindi $I(v_i, c_i) = v_i c_i$ se e solo se $v_i > 18$, 0 altrimenti.

**Esempio:** Per $v = [2, 14, 30, 24, 14]$ e $c = [6, 6, 9, 12, 6]$ la media pesata vale $\mu_{cv} = 30 * 9 + 24 * 12$

**A3.** Si scriva una funzione C `void mysplit(int * a, int dim)` *iterativa* che prenda in input un array di $n > 2$ interi $a$ contenente elementi *positivi* $a_0$, $a_1$, ..., $a_{n-1}$, e modifichi l'array originale associando all'elemento $i-$esimo la seguente formula:

$$a_i = \sum_{j=0}^{\lfloor p_i \rfloor} a_j a_{j+1} + \prod_{j=\lfloor p_i \rfloor}^{n-1} a_j$$

Nel calcolo di $a_i$ il punto di fulcro viene definito da $p_i = \min\left(\sum_{j=i}^{n-1} a_j, n-2\right)$.

**Esempio:** $a = [2, 1, 1, 1]$ quindi $n = 4$

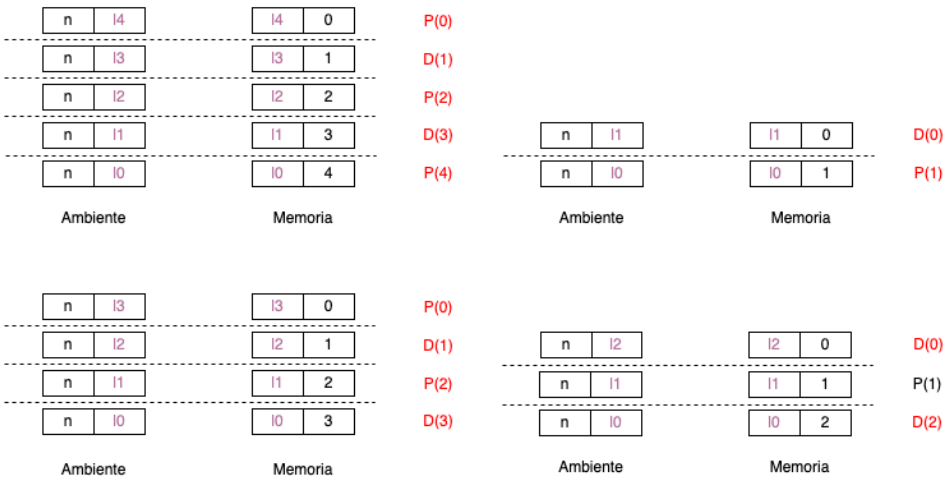


Figure 2.1. Esempio di computazione, in senso orario, di $P(4)$, $P(1)$, $D(3)$ e $D(2)$. La linea tratteggiata rappresenta un frame.

- per $i = 0$ vale $p_0 = \min(2 + 1 + 1 + 1, 4 - 2) = 2$ ed $a_0$ viene modificato con $a_0a_1 + a_1a_2 + a_2a_3 + a_2a_3$, e lo stesso valore viene calcolato per $i \in \{1, 2\}$ essendo $p_1 = p_2 = p_0$;
- per $i = 3$ vale $p_3 = \min(1, 4 - 2) = 1$ ed $a_3$ viene modificato con $a_0a_1 + a_1a_2 + a_1a_2a_3$.

## 2 Esercizi opzionali

### 2.1 Es. B1 (3 punti)

Si consideri la coppia ambiente/memoria e le chiamate associate (in rosso) nella figura qui sopra, per una computazione che utilizza due funzioni *mutualmente ricorsive* per valutare se un numero $n$ sia "pari" (funzione `P`) o "dispari" (funzione `D`). Definire, in C o Python, le due funzioni al fine di ottenere la memoria ed il flusso di esecuzione riportati.

**Nota:** due funzioni $f()$ e $g()$ sono mutualmente ricorsive se una chiama l'altra e viceversa.

### 2.2 Es. B2 (4 punti)

Si consideri questo programma C

```c
int my_op(int x)
{
  int y =4; // B
  return(x * y);
}

int main(void)
{
  int y = 6;

  for(int x = 1; x < y; y--)
  {
    // A
    int i = my_op(x);
    x = i;
  }
  // C
}
```

Si rappresenti la memoria del programma ai punti `A`, `B`, `C` per tutte le iterazioni del programma.

## 2.3 Es. B3 (5 punti)

Si consideri la seguente classe Python

```
class StrutturaSnellita():

    def __init__(self, dati, n, salto):
        self.dati = dati
        self.n = n
        self.salto = salto

    def __iter__(self):
        return IteratoreStruttura(self.dati, self.m, self.salto)
```

che viene utilizzata in questa maniera

```
print(list(StrutturaSnellita([1,2,3,4,5], 1)))        # 1,2,3,4,5
print(list(StrutturaSnellita([1,2,3,4,5], 2)))        # 1,3,5
print(list(StrutturaSnellita([1,2,3,4,5], 3)))        # 1,4
print(list(StrutturaSnellita([1,2,3,4,5], 4)))        # 1,5
print(list(StrutturaSnellita([1,2,3,4,5], 5)))        # 1
```

ovvero itera sulla lista `dati` mediante un iteratore di classe `IteratoreStruttura`, con la particolarità di mostrare un elemento ogni `salto` (con `salto=1` si ha un iteratore classico). Si definisca la classe `IteratoreStruttura`