

Esercizio 1 (*punti 2 in prima approssimazione*)

Consideriamo il seguente codice C:

```
#include <stdio.h>
#define z 3
int x = 1;

void p1 (void);
void p2 (int);

main() { int x = 2;
        p1();
        p2(x);
    }

void p1(void) { printf ("%d \n", x); }

int y = z+1;

void p2(int a) { printf ("%d %d \n", a, y); }
```

Domanda: qual'è l'output ?

Esercizio 2 (*punti 3 in prima approssimazione*)

Consideriamo il seguente codice C:

```
#include <stdio.h>

void cambia (int * x, int * y);

main ( )
{ int a = 2;
  int b = 4;
  cambia (&a, &b);
  printf ("%d , %d", a, b);
}

void cambia (int * x, int * y)
{ y = x;
  *y = 3;
}
```

Domanda: qual'è l'output ?

Esercizio 3 (punti 3 in prima approssimazione)

- 1) $2n^4 - 3n^3 + 18 \in \Omega(n^2)$? (risposta : sí / no)
- 2) $2^{n+3} \in O(2^n)$? (risposta : sí / no)
- 3) sapendo che
per ogni $n \geq 1000$: $f(n) \geq 0$, $g(n) \geq 0$, $|f(n) - g(n)| \leq 10$
posso concludere che $f \in \Theta(g)$? (risposta : sí / no)
(qui $|a|$ è il valore assoluto di a)

Esercizio 4 (punti 2 in prima approssimazione)

Consideriamo espressioni composte da: lettere, #, parentesi tonde.

L'insieme Exp delle espressioni è dato dalla seguente definizione (induttiva), dove:

Lett = {A,B,C,..., Z}, l'insieme delle lettere maiuscole.

- (base) Lett \subset Exp
- (passo_1) se $x, y \in$ Lett allora $x \# y \in$ Exp
- (passo_2) se $e1, e2 \in$ Exp allora $(e1) \# (e2) \in$ Exp

Domanda_1: Scrivere una stringa che appartiene ad Exp, di lunghezza ≥ 8

Domanda_2: (A)#(B) appartiene ad Exp ?

Esercizio 5 (punti 3 in prima approssimazione)

Consideriamo l'insieme Codici fatto da stringhe composte da

le 4 lettere maiuscole "DISI", 4 cifre binarie, 2 lettere maiuscole e precisamente della forma DISI $b_1 b_2 b_3 b_4 x y$

con x, y in {A, B, C, D,} e $b_1, b_2 \dots$ in {0, 1}.

Vogliamo rappresentare sottinsiemi (finiti) dell'insieme Codici usando delle tabelle hash con 50 buckets (o liste).

Per chi ha visto la faccenda con una distinzione tra elementi e chiavi:
qui elementi = chiavi.

Dovete definire una funzione di hashing h per fare ciò. Precisamente:

- 1) specificare la funzionalità di h , cioè dominio e codominio (cioè, h : \rightarrow )
- 2) definire h (cioè, $h(s) = \dots$)

Si chiede una funzione non stupida, ma non si pretende una funzione particolarmente astuta o complicata.

Esercizio 6 (*punti 1 in prima approssimazione*)

Il disegno che segue, rappresenta l'implementazione a "figlio sinistro, fratello destro" di un albero. Sul foglio risposte, disegnare l'abero corrispondente "nella maniera solita".

Esercizio 7 (punti 6 in prima approssimazione)

Considerate il "pezzo di programma" che segue (e che non fa nulla di interessante), dove: aa è un array $[1..n]$ of integer e j, k sono variabili intere e usiamo le $\{ \}$ come in C.

Pezzo di programma

```
(1)  j ← 2
(2)  while j < n do
      { (2.1)  k ← 3
        (2.2)  while k ≤ j do
                  { (2.2.1)  scrivi ( aa[ k ] )
                    (2.2.2)  k ← k+1
                  }
        (2.3)  vai a capo nell'output
        (2.4)  j ← j+1
      }
```

Domande.

- Dire qual'è l'output prodotto se $n = 8$ e $aa = 0, 1, 2, 3, 4, 5, 6, 7$
- Determinare la complessità del pezzo di programma (con un n generico), nel caso peggiore, possibilmente in $\Theta(\dots)$.

Non fare conti troppo dettagliati (esplicitando tutte le costanti,...),
ma non limitarsi nemmeno a dare il risultato, o a quattro chiacchiere;

in particolare: precisare se c'è un caso peggiore (o caso pessimo) e qual'è.

Esercizio 8 (punti 8 in prima approssimazione)

Si consideri la seguente procedura (dove usiamo le { } come in C):

procedura somma (aa : array [1..n] di interi, passato per riferimento;
sin, des : interi che servono da indici)

dichiarazioni: sum, centro, k variabili intere

istruzioni:

if sin ≤ des **then**

{ sum ← 0

per k = sin, sin+1,..., des : sum ← sum + aa[k]

equivale al Pascal: **for** k:= sin **to** des **do** sum := sum + aa[k]

scrivi (sum) e vai a capo sull'output

centro ← (sin + des) div 2

somma(aa, sin, centro-1)

somma(aa, centro+1, des)

}

Domande.

- Dire qual'è l'output prodotto dalla chiamata somma(aa, 1, 6) se n = 6 e aa = 1, 1, 1, 1, 1, 1
- Determinare la complessità della generica chiamata somma(aa, 1, n) (con n generico) nel caso peggiore, possibilmente in $\Theta(\dots)$.

Non fare conti troppo dettagliati (esplicitando tutte le costanti,...),
ma non limitarsi nemmeno a dare il risultato, o a quattro chiacchiere;

in particolare, precisare se c'è un caso peggiore (o caso pessimo) e qual'è.