
```
int mistero(int a, int b)
{
  if (b == 0) return a;
  return mistero(a / 2, b - 1);
}
```

Figure 1: Esempio di pseudocodice

Rispondere alle domande a risposta multipla annerendo la casella corrispondente alla risposta corretta. Ogni domanda ha una ed una sola risposta corretta.

Cognome e Nome: .....

Matricola: .....

Domanda 1 La funzione implementata dallo pseudo-codice di Figura 1:

- ☐ Può causare una crescita incontrollata dello stack
- ☐ Causa sempre ricorsione infinita
- ☐ Nessuna delle altre risposte
- ☐ Non può essere implementata per via iterativa
- ☐ Non usa ricorsione in coda

Domanda 2 β-riducendo $((\lambda a.aaa)(\lambda b.b))(\lambda c.c)$ si ottiene:

- ☐ $\lambda x.xa$
- ☐ La riduzione non termina
- ☐ aaa
- ☐ $\lambda c.c$
- ☐ Nessuna delle altre risposte

Domanda 3 Dato il frammento di programma (espresso in pseudo-codice) contenuto in Figura 2, qual'è il valore di ritorno di `topolino()`, assumendo scope statico?

- ☐ Nessuna delle altre risposte
- ☐ Non è possibile dirlo
- ☐ 0
- ☐ -3
- ☐ 11

Domanda 4 β-riducendo $(\lambda n.\lambda m.\lambda f.\lambda x.(nf)((mf)x))(\lambda f.\lambda x.fx)(\lambda f.\lambda x.x)$ si ottiene:

- ☐ $\lambda f.\lambda x.f(f(f(fx)))$
- ☐ $\lambda f.\lambda x.fx$
- ☐ x
- ☐ Nessuna delle altre risposte
- ☐ fx

```
int x, y;

void pippo(void)
{
  x = 8;
  y = 4;
}

void pluto(void)
{
  int y;

  pippo();
  y = 3;
}

int topolino(void)
{
  int x, z;

  x = 5;
  y = 15;
  z = x + y;
  pluto();

  return z - y - x;
}
```

Figure 2: Esempio di pseudocodice

Domanda 5 I record di attivazione:

- ☐ Sono necessari solo in presenza di funzioni di ordine superiore
- ☐ Sono allocati dinamicamente solo in caso di scope dinamico
- ☐ Nessuna delle altre risposte
- ☐ Devono essere esplicitamente allocati e deallocati dal codice del programma che li usa
- ☐ Sono allocati solo nello heap

Domanda 6 Dato il frammento di programma (espresso in pseudo-codice) contenuto in Figura 3, qual'è il valore di ritorno di `f1()`, assumendo scope dinamico?

- ☐ Nessuna delle altre risposte
- ☐ -1
- ☐ -5
- ☐ 5
- ☐ Non è possibile dirlo

Domanda 7 Si consideri lo pseudo-codice di Figura 4. Qual'è il valore di ritorno di `pluto()` se i parametri sono passati *per valore*?

- ☐ Nessuna delle altre risposte
- ☐ Dipende dal tipo di scope (statico o dinamico) utilizzato
- ☐ Non è possibile passare `c + 1` per valore
- ☐ 6
- ☐ 10

```
int x, y, z;

void f3(void)
{
  x = 0;
  y = 5;
}

void f2(void)
{
  int y;

  f3();
  y = 0;
  z = 10;
}

int f1(void)
{
  int x;

  x = -5;
  y = 10;
  z = x + y;
  f2();

  return z - y - x;
}
```

Figure 3: Esempio di pseudocodice

```
int c = 2;

int pippo(int a)
{
  c = c + 2;
  return a * 2;
}

int pluto(void)
{
  return(pippo(c + 1));
}
```

Figure 4: Esempio di pseudocodice

```
int c = 2;

int pippo(int a)
{
  c = c + 2;
  return a * 2;
}

int pluto(void)
{
  return(pippo(c + 1));
}
```

Figure 5: Esempio di pseudocodice

Domanda 8 Si consideri lo pseudo-codice di Figura 5. Qual'è il valore di ritorno di `pluto()` se i parametri sono passati *per nome*?

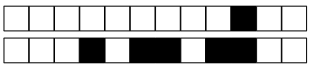
- ☐ 10
- ☐ Nessuna delle altre risposte
- ☐ 6
- ☐ 4
- ☐ Dipende dal tipo di scope (statico o dinamico) utilizzato

Domanda 9 La memoria gestita staticamente:

- ☐ E' allocata dal compilatore prima dell'esecuzione del programma. Le entità allocate staticamente in memoria risiedono in una zona fissa di memoria durante tutta l'esecuzione del programma
- ☐ Nessuna delle altre risposte
- ☐ E' allocata esplicitamente dal programma a tempo di esecuzione, ma una volta allocata è staticamente legata al programma e non può essere liberata fino alla sua terminazione
- ☐ E' una memoria a sola lettura
- ☐ E' allocata prima dell'esecuzione del programma. Le entità allocate staticamente possono essere deallocate durante l'esecuzione del programma, per liberare memoria

Domanda 10 Si può dire che una macchina astratta che capisce il linguaggio C non sia implementata in modo puramente compilativo perché:

- ☐ Gli eseguibili generati da un compilatore C in genere non eseguono direttamente sulla macchina hardware, ma su una macchina astratta che include il *runtime* del linguaggio e le funzionalità del Sistema Operativo
- ☐ Gli eseguibili generati dal compilatore vengono comunque interpretati da una macchina virtuale
- ☐ Una macchina astratta che capisca un linguaggio di alto livello come il C non è mai implementabile con un compilatore
- ☐ Il *runtime* del linguaggio C è comunque sempre interpretato
- ☐ Nessuna delle altre risposte



Domanda 11 Se gli array sono memorizzati *per colonne* e `char a[100][100]` è un array multidimensionale di caratteri con `a[0][0]` che ha indirizzo `0x1100`, qual'è l'indirizzo di `a[5][10]`?:

- ☐ 0x21FE
- ☐ Nessuna delle altre risposte
- ☐ 0x22FE
- ☐ 0x24ED
- ☐ 0x14ED

Domanda 12 Il costrutto `for` dei linguaggi `C`, `C++` e `Java`:

- ☐ Non è un costrutto di iterazione determinata
- ☐ E' necessario a tali linguaggi per implementare qualsiasi tipo di algoritmo
- ☐ Nessuna delle altre risposte
- ☐ E' un costrutto di iterazione determinata
- ☐ Permette di sapere in anticipo quante volte il ciclo verrà ripetuto (indipendentemente dal corpo del ciclo)