```
int b = 666;

int pippo(int x)
{
  x = 666; b = 1;
  return x / 2;
}

int pluto(void)
{
  int a, c;

  a = b / 333;
  c = pippo(a);

  return a + c;
}
```

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: .....

**Question 1** Un garbage collector:

- ☐ Richiede un'implementazione complessa, usando la tecnica dei *tombstone* (pietre tombali)
- ☐ E' implementabile solo in linguaggi di programmazione funzionali
- ☐ Può essere implementato tramite la tecnica del reference counting (contatore dei riferimenti), che riesce sempre ad identificare tutta la memoria allocata dinamicamente ma non più utilizzata
- ☐ E' implementabile tramite la tecnica di lucchetti e chiavi, che però può causare dei memory leak
- ☒ Nessuna delle altre risposte

**Question 2** Si consideri lo pseudo-codice di Figura 1. Qual'è il valore di ritorno di `pluto()` se i parametri sono passati *per valore*?

- ☒ Nessuna delle altre risposte
- ☐ 666
- ☐ 999
- ☐ Dipende dal tipo di scope (statico o dinamico) utilizzato
- ☐ 333

```
int somma(int a, int b)
{
  if (a == 0) return b;
  return somma(a - 1, b) + 1;
}
```

Figure 2: Esempio di pseudocodice

**Question 3** Funzione implementata dallo pseudo-codice di Figura 2:

- [ ] Nessuna delle altre risposte
- [ ] Non può essere implementata per via iterativa
- [ ] Causa sempre ricorsione infinita
- [ ] Usa ricorsione in coda
- [x] Non usa ricorsione in coda

**Question 4** Se gli array sono memorizzati *per righe* e `char a[100][100]` è un array multidimensionale di caratteri con `a[0][0]` che ha indirizzo `0x2000`, qual'è l'indirizzo di `a[5][10]`?:

- [ ] 0x200F
- [x] 0x21FE
- [ ] Nessuna delle altre risposte
- [ ] 0x23ED
- [ ] 0x2510

**Question 5** $\beta$-riducendo $(\lambda n.\lambda m.\lambda f.\lambda x.(nf)((mf)x))(\lambda f.\lambda x.fx)(\lambda f.\lambda x.x)$ si ottiene:

- [ ] Nessuna delle altre risposte
- [ ] $x$
- [ ] $\lambda f.\lambda x.fffffffx$
- [x] $\lambda f.\lambda x.fx$
- [ ] $fx$
- [ ] $\lambda f.\lambda x.fffffx$

**Question 6** Dato il frammento di programma (espresso in pseudo-codice) contenuto in Figura 3, qual'è il valore di ritorno di `topolino()`, assumendo scope statico?

- [ ] Non è possibile dirlo
- [ ] 0
- [ ] -3
- [x] 11
- [ ] Nessuna delle altre risposte

**Question 7** $\beta$-riducendo $(\lambda a.((a\lambda b.\lambda c.c)\lambda d.\lambda e.d))(\lambda f.\lambda g.g)$ si ottiene:

- [ ] Nessuna delle altre risposte
- [x] $\lambda b.\lambda c.b$
- [ ] $b$
- [ ] $\lambda b.\lambda c.c$
- [ ] La riduzione non termina

```
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 3: Esempio di pseudocodice

**Question 8** Si consideri lo pseudo-codice di Figura 4. Qual'è il valore di ritorno di `pluto()` se i parametri sono passati *per nome*?

- [ ] Non è possibile passare `a` per nome
- [ ] Dipende dal tipo di scope (statico o dinamico) utilizzato
- [ ] Nessuna delle altre risposte
- [ ] 335
- [x] 999

**Question 9** Un *interprete* di un linguaggio $\mathcal{L}$ scritto in un linguaggio $\mathcal{L}_O$ è:

- [ ] Nessuna delle altre risposte
- [ ] Un programma che trasforma un programma $P^{\mathcal{L}}$ (espresso nel linguaggio $\mathcal{L}$) in un programma $P^{\mathcal{L}_O}$ (espresso nel linguaggio $\mathcal{L}_O$) tale che per ogni input $I$ si ha $P^{\mathcal{L}}(I) = P^{\mathcal{L}_O}(I)$
- [ ] L'implementazione di una macchina astratta scritta nel linguaggio $\mathcal{L}_O$, che capisce programmi scritti nel linguaggio $\mathcal{L}$
- [x] Un programma scritto nel linguaggio $\mathcal{L}_O$ che riceve come ingresso un programma $P^{\mathcal{L}}$ (espresso nel linguaggio $\mathcal{L}$) ed il suo input $I$ generando lo stesso output che genera $P^{\mathcal{L}}$ con input $I$
- [ ] Una implementazione di macchine astratte indipendente dalla macchina fisica

```
int b = 666;

int pippo(int x)
{
  x = 666; b = 1;
  return x / 2;
}

int pluto(void)
{
  int a, c;

  a = b / 333;
  c = pippo(a);

  return a + c;
}
```

Figure 4: Esempio di pseudocodice

**Question 10** La memoria gestita dinamicamente:

- [x] E' necessaria per implementare la ricorsione
- [ ] Nessuna delle altre risposte
- [ ] Non è mai strettamente necessaria, ma permette di ottenere migliori prestazioni
- [ ] E' necessaria solo per implementare macchine astratte per linguaggi funzionali
- [ ] E' usata solo in linguaggi interpretati

**Question 11** In caso di *scope dinamico*:

- [ ] Il valore assegnato ad una variabile non può essere modificato
- [ ] Non è possibile annidare più blocchi di istruzioni
- [ ] Nessuna delle altre risposte
- [x] I legami fra nomi ed oggetto possono essere determinati solo a tempo di esecuzione
- [ ] I legami fra nomi ed oggetto possono essere determinati semplicemente leggendo il testo di un programma

**Question 12** $\beta$-riducendo $(\lambda a.aab)((\lambda a.aab)(\lambda a.(\lambda b.ba)c))$ si ottiene:

- [ ] La riduzione non termina
- [ ] $aab$
- [ ] $\lambda a.aab$
- [ ] $ccb(ccb)c$
- [x] Nessuna delle altre risposte