```
int c = 2;

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

int pluto(void)
{
  return(pippo(c + 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: .....

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

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

Question 2 β-riducendo $(\lambda n.\lambda m.\lambda f.\lambda x.(nf)((mf)x))(\lambda f.\lambda x.fffffx)(\lambda f.\lambda x.fx)$ si ottiene:

- ☐ $\lambda f.\lambda x.fffffx$
- ☐ f
- ☐ Nessuna delle altre risposte
- ☐ $\lambda f.\lambda x.fffffffx$
- ☐ x
- ☐ $\lambda f.\lambda x.ffffffx$

Question 3 Un *compilatore* da un linguaggio $\mathcal{L}$ ad un linguaggio $\mathcal{L}_O$ è:

- ☐ 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 $P^{\mathcal{L}}(I) = P^{\mathcal{L}_O}(I)$
- ☐ Nessuna delle altre risposte
- ☐ 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
- ☐ L'implementazione di una macchina astratta scritta nel linguaggio $\mathcal{L}_O$, che capisce programmi scritti nel linguaggio $\mathcal{L}$
- ☐ Una implementazione di macchine astratte indipendente dalla macchina fisica

Question 4 L'allocazione dinamica della memoria:

- ☐ Nessuna delle altre risposte
- ☐ E' sempre effettuata solo dal compilatore o dall'interprete
- ☐ Può essere fatta solo dallo *heap*
- ☐ Può essere fatta solo dallo *stack*
- ☐ Può essere fatta sia dallo *stack* che dallo *heap*

Question 5 β-riducendo $(\lambda n.\lambda f.\lambda x.f((nf)x))(\lambda f.\lambda x.fffffx)$ si ottiene:

- ☐ Nessuna delle altre risposte
- ☐ $\lambda f.\lambda x.fffffffx$
- ☐ La riduzione non termina
- ☐ $\lambda f.\lambda x.ffffffx$
- ☐ fx
- ☐ L'espressione è irriducibile

Question 6 La memoria gestita staticamente:

- ☐ 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' allocata prima dell'esecuzione del programma. Le entità allocate staticamente possono essere deallocate durante l'esecuzione del programma, per liberare memoria
- ☐ 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
- ☐ E' una memoria a sola lettura
- ☐ Nessuna delle altre risposte

Question 7 In caso di *scope statico*:

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

Question 8 Se gli array sono memorizzati per righe ed `int a[100][100]` è un un array multidimensionale di interi (si assuma che la dimensione di un intero sia 4 byte) con `a[0][0]` che ha indirizzo `0x10000`, qual'è l'indirizzo di `a[5][10]`?:

- ☐ 0x100F
- ☐ Nessuna delle altre risposte
- ☐ 0x13ED
- ☐ 0x11FE
- ☐ 0x1510

```
int a, b, c;

void pippo(void)
{
  int a;

  a = 6;
  b = 5;
}

void pluto(void)
{
  int c;
  int b;

  pippo();
  c = 3;
  a = 4;
}

void topolino(void)
{
  int a;

  a = 1;
  b = 10;
  pluto();

  c = a + b;
}
```

Figure 2: Esempio di pseudocodice

Question 9 L'*ambiente* (o *environment*) è:

- ☐ Un insieme di associazioni (nome, valore) definite staticamente durante lo sviluppo di un programma
- ☐ Una lista di coppie (nome, tipo) che permette di accedere alle variabili di un programma
- ☐ Nessuna delle altre risposte
- ☐ L'insieme dei valori che una variabile assume durante l'esecuzione di un programma
- ☐ L'insieme delle associazioni (nome, oggetto denotabile) esistenti in uno specifico punto del programma ed in uno specifico momento durante l'esecuzione di un programma

Question 10 Dato il frammento di programma (espresso in pseudo-codice) della Figura 2, quanto vale la variabile globale `c` dopo aver eseguito `topolino()`, assumendo scope statico?

- ☐ Nessuna delle altre risposte
- ☐ 6
- ☐ Non è possibile dirlo
- ☐ 14
- ☐ 3

```
int c = 2;

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

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

Figure 3: Esempio di pseudocodice

Question 11 La ricorsione in coda:

- ☐ Non è implementabile nei linguaggi imperativi
- ☐ Nessuna delle altre risposte
- ☐ Richiede di non scrivere mai la chiamata ricorsiva come ultimo statement di una subroutine
- ☐ Richiede di non ritornare mai direttamente il valore ritornato da una chiamata ricorsiva
- ☐ Permette di risolvere il problema della ricorsione infinita

Question 12 Si consideri lo pseudo-codice di Figura 3. Qual'è il valore di ritorno di `pluto()` se i parametri sono passati *per valore*?

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