Prova di esame del 23 gennaio 2017

Esercizio 1) [10 punti]

Marcare le affermazioni che si ritengono vere. Ogni domanda può avere un qualunque numero naturale di affermazioni vere. Vengono **assegnati** 0.5 punti sia per ogni affermazione *vera che viene marcata* che per ogni affermazione *falsa che viene lasciata non marcata*. Analogamente vengono **sottratti** 0.5 punti sia per ogni affermazione *falsa che viene marcata* che per ogni affermazione *vera che viene lasciata non marcata*.

1. …
   a. Una classe *deferred* può ereditare da una classe *effective*
   b. Una classe *C* non può ereditare da due classi differenti *B1* e *B2*, se sia *B1* che *B2* hanno una stessa classe *A* come antenato comune
   c. Ad un entità di tipo statico *C* non si può assegnare un oggetto di un tipo che è un antenato di *C*
   d. In una classe *C* una feature *f* ereditata dalla classe *A* può essere ridefinita soltanto se *f* è *deferred* in A

2. Una query …
   a. … può essere usata come procedura di creazione
   b. … può essere implementata come una *routine*
   c. … può apparire nelle precondizioni e postcondizioni di una qualunque routine
   d. … può apparire nell'invariante di classe

3. …
   a. Una funzione è una query implementata mediante memorizzazione
   b. Un attributo è una query implementata mediante memorizzazione
   c. Una routine è l'implementazione di una feature mediante computazione
   d. Un attributo è l'implementazione di una feature mediante computazione

4. …
   a. Il risultato ritornato da una query è fornito sempre da una funzione
   b. Il risultato ritornato da una query è fornito sempre da un attributo
   c. Il risultato ritornato da una query è fornito da una funzione o da un attributo
   d. Il risultato ritornato da un comando è fornito da una procedura o da un attributo

5. …
   a. Se *C* è una classe *deferred* allora non possono esistere nel programma entità di tipo statico *C*
   b. Se *C* è una classe che usa la *feature* di un'altra classe S allora C è *supplier* (fornitore) di S
   c. La classe *C* può essere contemporaneamente un *supplier* (fornitore) ed un *client* (cliente)
   d. Se il tipo di una variabile è un tipo espanso allora il valore della variabile a tempo di esecuzione (*run-time*) è un oggetto

SOLUZIONE:

1. …
   a. Una classe *deferred* può ereditare da una classe *effective*
   b. Una classe *C* non può ereditare da due classi differenti *B1* e *B2*, se sia *B1* che *B2* hanno una stessa classe *A* come antenato comune
   c. Ad un entità di tipo statico *C* non si può assegnare un oggetto di un tipo che è un antenato di *C*
   d. In una classe *C* una feature *f* ereditata dalla classe *A* può essere ridefinita soltanto se *f* è *deferred* in A

2. Una query …
   a. … può essere usata come procedura di creazione
   b. … può essere implementata come una *routine*
   c. … può apparire nelle precondizioni e postcondizioni di una qualunque routine
   d. … può apparire nell'invariante di classe

3. …
   a. Una funzione è una query implementata mediante memorizzazione
   b. Un attributo è una query implementata mediante memorizzazione
   c. Una routine è l'implementazione di una feature mediante computazione
   d. Un attributo è l'implementazione di una feature mediante computazione

4. …
   a. Il risultato ritornato da una query è fornito sempre da una funzione
   b. Il risultato ritornato da una query è fornito sempre da un attributo
   c. Il risultato ritornato da una query è fornito da una funzione o da un attributo
   d. Il risultato ritornato da un comando è fornito da una procedura o da un attributo

5. …
   a. Se *C* è una classe *deferred* allora non possono esistere nel programma entità di tipo statico *C*
   b. Se *C* è una classe che usa la *feature* di un'altra classe S allora C è *supplier* (fornitore) di S
   c. La classe *C* può essere contemporaneamente un *supplier* (fornitore) ed un *client* (cliente)
   d. Se il tipo di una variabile è un tipo espanso allora il valore della variabile a tempo di esecuzione (*run-time*) è un oggetto

Esercizio 2) [10 punti]

La classe *INT_LINKABLE* modella un elemento di una lista che può rappresentare valori interi. La sua implementazione è la seguente:

```
class
   INT_LINKABLE
create
   make

feature -- accesso
   value : INTEGER
      -- L'intero memorizzato in questo elemento

   next : INT_LINKABLE
      -- Il successivo elemento della lista

feature -- operazioni fondamentali
   make (a_value : INTEGER)
      -- crea l'elemento
      do
         value := a_value
      ensure
         value = a_value
      end

   link_to (other: INT_LINKABLE)
        -- collega questo elemento con `other'
      do
         next := other
      ensure
         next = other
      end

   link_after (other: INT_LINKABLE)
        -- inserisce questo elemento dopo `other' conservando quello che c'era dopo di esso
      require
         other /= Void
      do
         link_to (other.next)
         other.link_to (Current)
      ensure
         other.next = Current
         other.next.next = old other.next
      end

end
```

La classe *INT_LINKED_LIST* modella una lista di interi. Una parte della sua implementazione è fornita qua sotto:

```
class
   INT_LINKED_LIST

feature -- accesso
   first_element: INT_LINKABLE
        -- Il primo elemento della lista

   last_element: INT_LINKABLE
        -- L'ultimo elemento della lista

   count: INTEGER
        -- Il numero di elementi della lista
```

Aggiungere alla classe *INT_LINKED_LIST* una feature *insert_multiple_before* (*a_value*, *target*: *INTEGER*) che inserisce un nuovo elemento di valore *a_value* **subito prima di ogni** occorrenza di *target*. Altrimenti il nuovo elemento viene inserito all'inizio della lista.

Indicare anche contratti ed eventuali invarianti. Non è possibile modificare la classe *INT_LINKABLE*. Il numero di righe vuote non ha alcuna relazione col numero di istruzioni/dichiarazioni da scrivere.

```
feature -- operazioni fondamentali
   insert_multiple_before (a_value, target: INTEGER): BOOLEAN
        -- Inserisce `a_value' subito prima di ogni occorrenza di `target'
        -- Altrimenti inserisce `a_value' all'inizio

      require
         ____________________________________________________________________
         ____________________________________________________________________
      local
         ____________________________________________________________________
         ____________________________________________________________________
      do
         ____________________________________________________________________
         ____________________________________________________________________
         ____________________________________________________________________
         ____________________________________________________________________
         ____________________________________________________________________
         ____________________________________________________________________
         ____________________________________________________________________
         ____________________________________________________________________
         ____________________________________________________________________
         ____________________________________________________________________
         ____________________________________________________________________
      ensure
         ____________________________________________________________________
         ____________________________________________________________________
end
```

SOLUZIONE:

```
feature -- operazioni fondamentali
  insert_multiple_before (a_value, target: INTEGER): BOOLEAN
       -- Inserisce `a_value' subito prima di ogni occorrenza di `target'
       -- Altrimenti inserisce `a_value' all'inizio
     local
       previous_element, current_element, new_element: like first_element
     do
       if has (target) then
         from
           previous_element := Void
           current_element := first_element
         until
           current_element = Void
         loop
           if current_element.value = target then
             create new_element.make (a_value)
             if previous_element = Void then
               new_element.link_to(first_element)
               first_element := new_element
             else
               previous_element.link_to(new_element)
               new_element.link_to(current_element)
             end
             count := count + 1
           end
           previous_element := current_element
           current_element := current_element.next
         end
       else -- la lista non contiene `target'
         create new_element.make (a_value)
         if count = 0 then
           first_element := new_element
           last_element := first_element
         else
           new_element.link_to(first_element)
           first_element := new_element
         end -- loop
         count := count + 1
       end
     ensure
       di_piu: count > old count
       in_testa_se_non_presente: not (old has (target)) implies first_element.value =
a_value
       collegato_se_presente: old has (target) implies get_element(a_value).next.value =
target
  end
```

Si noti che l'ultima post condizione garantisce che sia stato fatto l'inserimento di *a_value* soltanto prima della prima occorrenza di *target* nella lista.

Esercizio 3) [10 punti]
Completare i contratti (pre-condizioni, post-condizioni, invarianti di classe) della classe *CRUISE_CONTROL* sotto descritta che modella un sistema software per il controllo della velocità di crociera di un veicolo in modo da rispecchiare la seguente specifica informale:

1. Si può operare sul veicolo con i seguenti comandi: premere sui freni, premere sull'acceleratore, rilasciare l'acceleratore. In corrispondenza di tali comandi sul veicolo, il sistema software riceve i comandi, rispettivamente, *brake*, *increase_gas*, *decrease_gas*.
2. Il sistema software di controllo della velocità di crociera (cruise control = CC) è in uno dei seguenti stati: *disattivo*, *attivo*, *abilitato* (cioè attivo con controllo della velocità in corso), *disabilitato* (cioè attivo ma senza controllo della velocità).
3. Il comando di abilitazione (*enable_CC*) richiede che sia stata impostata una velocità di crociera (*cruise_speed*) > 0.
4. La velocità può essere impostata o re-impostata (*set_speed*) in qualunque momento CC è attivo. La disattivazione di CC causa la de-impostazione della velocità di crociera.
5. In stato *abilitato* la pressione sui freni (*brake*) o la pressione dell'acceleratore (*increase_gas*) causano la disabilitazione di CC.
6. In stato *disabilitato* il rilascio dell'acceleratore (*decrease_gas*) causa l'abilitazione di CC.

Non c'è relazione tra il numero di righe vuote ed il numero di contratti da scrivere. Non è necessario conoscere altro codice all'infuori di quello fornito.

Laddove si ritiene che non abbia senso inserire pre- o post-condizioni le righe dei relativi contratti **devono essere sbarrate**.

```
deferred class
   CRUISE_CONTROL

feature {ANY}   -- accesso
   cruise_speed : INTEGER
      -- La velocità di crociera che è stata impostata. Vale 0 quando non è stata impostata.

feature {ANY}   -- stato
   is_CC_on : BOOLEAN
      -- Il sistema CC è attivo?
   is_CC_enabled : BOOLEAN
      -- Il sistema CC è abilitato?
      require
         is_CC_on
      deferred
      ensure
         Result = (cruise_speed /= 0)
      end

feature {ANY}   -- operazioni sul veicolo
   brake
      -- Sono stati premuti i freni
      require
         ____________________________________________________________________________

         ____________________________________________________________________________

      deferred
      ensure
         ____________________________________________________________________________

         ____________________________________________________________________________

      end
```

```
   increase_gas
      -- È stato premuto l'acceleratore
      require
         _____________________________________________________________________________
         _____________________________________________________________________________
      deferred
      ensure
         _____________________________________________________________________________
         _____________________________________________________________________________
      end

   decrease_gas
      -- È stato rilasciato l'acceleratore
      require
         _____________________________________________________________________________
         _____________________________________________________________________________
      deferred
      ensure
         _____________________________________________________________________________
         _____________________________________________________________________________
      end


feature {ANY}  -- operazioni sul sistema di controllo
   set_speed (a_speed : INTEGER)
      -- Assegna la velocità di crociera.
      require
         _____________________________________________________________________________
         _____________________________________________________________________________
      deferred
      ensure
         _____________________________________________________________________________
         _____________________________________________________________________________
      end

   enable_CC
      -- Abilita il controllo della velocità
      require
         _____________________________________________________________________________
         _____________________________________________________________________________
      deferred
      ensure
         _____________________________________________________________________________
         _____________________________________________________________________________
      end
```

```
   disable_CC
      -- Disabilita il sistema CC.
      require
         _____________________________________________________________________________
         _____________________________________________________________________________
      deferred
      ensure
         _____________________________________________________________________________
         _____________________________________________________________________________
      end

   switch_on
      -- Accende il sistema CC.
      require
         _____________________________________________________________________________
         _____________________________________________________________________________
      deferred
      ensure
         _____________________________________________________________________________
         _____________________________________________________________________________
      end

   switch_off
      -- Spegne il sistema CC.
         _____________________________________________________________________________
         _____________________________________________________________________________
      deferred
      ensure
         _____________________________________________________________________________
         _____________________________________________________________________________
      end

invariant
_____________________________________________________________________________
_____________________________________________________________________________
end
```

SOLUZIONE:

```
deferred class
   CRUISE_CONTROL

feature {ANY}  -- accesso
   cruise_speed : INTEGER
      -- La velocità di crociera che è stata impostata. Vale 0 quando non è stata impostata.

feature {ANY}  -- stato
   is_CC_on : BOOLEAN
      -- Il sistema CC è acceso?
```

```
   is_CC_enabled : BOOLEAN
      -- Il sistema CC è abilitato?
      require
         is_CC_on
      deferred
      ensure
         Result = (cruise_speed /= 0)
      end

feature {ANY}  -- operazioni sul veicolo
   brake
      -- Sono stati premuti i freni
      require
         -- l'operazione non ha prerequisiti
      deferred
      ensure
         not is_CC_enabled
      end

   increase_gas
      -- È stato premuto l'acceleratore
      require
         -- l'operazione non ha prerequisiti
      deferred
      ensure
         not is_CC_enabled
      end

   decrease_gas
      -- È stato rilasciato l'acceleratore
      require
         -- l'operazione non ha prerequisiti
      deferred
      ensure
         is_CC_enabled
      end


feature {ANY}  -- operazioni sul sistema
   set_speed (a_speed : INTEGER)
      -- Assegna la velocità di crociera
      require
         is_CC_on
         a_speed /= 0
      deferred
      ensure
         cruise_speed = a_speed
      end
```

```
   enable_CC
      -- Abilita il sistema CC
      require
         not is_CC_enabled
         cruise_speed /= 0
      deferred
      ensure
         is_CC_enabled
      end

   disable_CC
      -- Disabilita il sistema CC
      require
         is_CC_enabled
      deferred
      ensure
         not is_CC_enabled
      end

   switch_on
      -- Accende il sistema CC
      require
         not is_CC_on
      deferred
      ensure
         is_CC_on
         not is_CC_enabled
      end

   switch_off
      -- Spegne il sistema CC
      require
         is_CC_on
      deferred
      ensure
         not is_CC_on
         cruise_speed = 0
         not is_CC_enabled
      end

invariant
   current_speed ≥ 0
   is_CC_enabled implies is_CC_on

end
```

Si noti che per le tre feature relative ad operazioni sul veicolo (*brake*, *increase_gas*, *decrease_gas*) non ha senso inserire pre-condizioni dal momento che tali feature vengono invocate dall'esecuzione delle corrispondenti operazioni fisiche sul veicolo, operazioni che una volta attivate sul piano fisico devono essere necessariamente accettate e gestite dal sistema software.

Si noti anche, per le stesse tre feature, che le post-condizioni possono essere equivalentemente scritte, invece che come sopra illustrato, come **old** *X* **implies not** *X* oppure **old not** *X* **implies** *X*.