Esercizi di verifica: progettazione di casi di test usando criteri strutturali.

1. MyAir

Si consideri il metodo stimaLivelli, con la seguente specifica: dato un vettore di associati del club MyAir, restituisce una tripla formata, rispettivamente, dal numero degli associati che, in base alla loro situazione: migliorano il proprio livello, non lo modificano, lo peggiorano. Si noti che i livelli standard, argento e oro sono codificati rispettivamente con 1, 2, e 3.

```
public Tripla stimaLivelli(Vector <AssociatoMyAir> associatiMyAir) {
   int inAumento = 0;
   int stabili = 0;
   int inCalo = 0;
   for (AssociatoMyAir m : associatiMyAir) { // itera su associatiMyAir
     int livello = m.getLivello();
     int miglia  = m.getMiglia();
      int differenza;
     if (miglia > 100000)          differenza = 3 - livello;
       else if (miglia > 15000) differenza = 2 - livello;
            else                  differenza = 1 - livello;
     if (differenza > 0)           inAumento++;
       else if (differenza==0)     stabili++;
            else                  inCalo++;
   }
   return new Tripla(inAumento, stabili, inCalo);
}
```

Definire un insieme di coppie di attributi (livello, miglia) da attribuire agli elementi del vettore `associatiMyAir` che permetta di raggiungere una copertura del 100% del codice del metodo `stimaLivelli`, secondo il criterio dei comandi.

Risposta. Una soluzione minimale è la seguente:

Livello	Miglia
2	120000
3	75000
1	10000

Analizzare anche i criteri funzionali

2. Stub e Criteri strutturali

Il pedaggio si calcola considerando: la tariffa unitaria a chilometro, il tipo di veicolo utilizzato (5 classi), le caratteristiche dei tratti autostradali percorsi (di pianura o di montagna).

Si supponga il calcolo sia fatto usando i metodi così specificati:

```
      /*dati i caselli di ingresso e di uscita, restituisce il numero di km di pianura
      e il numero di quelli di montagna*/
int[ ] calcolaChilometri(a String, b String)
      /*dati i caselli di ingresso e di uscita e la classe del veicolo,
      ottiene il numero di Km percorsi e calcola il pedaggio */
double calcolaPedaggio(a String, b String, c Classe)
```

Per quale dei due metodi dati sopra potrebbe essere utile creare uno stub, nella verifica del calcolo del pedaggio? Definire un semplice stub, che permetta la ripetibilità dei test, e non sia banale (vari i risultati in funzione degli argomenti).

Risposta. Notando che calcolaPedaggio deve invocare calcolaChilometri, e che la realizzazione di questo metodo richiede l'accesso al DBrete, conviene testare calcolaPedaggio con uno stub per calcolaChilometri. Per permettere la ripetibilità dei test non si può usare un generatore di numeri pseudo-casuali. La soluzione che segue conserva la proprietà commutativa di calcolaChilometri (andando da A a B si fanno gli stessi chilometri che andando da B a A) e può produrre anche risultati estremi (tratto di montagna o di pianura lungo zero):

```
int[ ] calcolaChilometri(a String, b String){
      int[] coppia = {0,0};
      int mx = max(a.length(),b.length());
      int mn = min(a.length(),b.length());
      coppia[0]= mx – mn ;
      coppia[1]= (2*mn >= mx ? 2*mn–mx : mn) ;
      return coppia;}
```

3. continua ex precedente

Si supponga ora che il calcolo del pedaggio sia fatto usando il metodo così specificato:

```
/*dati i il numero di km di pianura p, il numero di km di montagna m e la classe del
veicolo c, calcola il pedaggio */


double calcolaPedaggio(p int, m int, c int) {
double pedaggio = 0.0;
      if (m <= SOGLIA_MONTAGNA) {
            pedaggio = (p + m) * c * COSTO_UNITARIO;
      } else {
            pedaggio = (p + 2*m) * c * COSTO_UNITARIO;
       }
      if (c<5) {
            pedaggio = pedaggio * (1 + ALIQUOTA_IVA);
        }
      return pedaggio;
}
```

a) Definire i valori di input di un insieme di casi di prova per la convalida del metodo *calcolaPedaggio*, applicando un criterio di progettazione delle prove a *scatola aperta*.

L'ambiente di prova prevede i seguenti valori per le costanti usate dal metodo:

```
    private final double ALIQUOTA_IVA          = 0.20;
    private final double COSTO_UNITARIO  = 0.20;
    private final double SOGLIA_MONTAGNA = 50;
```

b) perchè, in base alle informazioni fornite nel testo, non è possibile completare i casi di prova?

Risposta.

a) Una possibile selezione di casi di prova, relativamente ai valori di input, che permette una copertura del 100% secondo il criterio delle decisioni è data dalla seguente tabella:

b) Input		
c) P	d) M	e) c
f) 10	g) 40	h) 1
i) 10	j) 60	k) 5

b) Il completamento richiederebbe la specifica dei risultati attesi in output, ma le informazioni nel testo non permettono di costruire un oracolo.

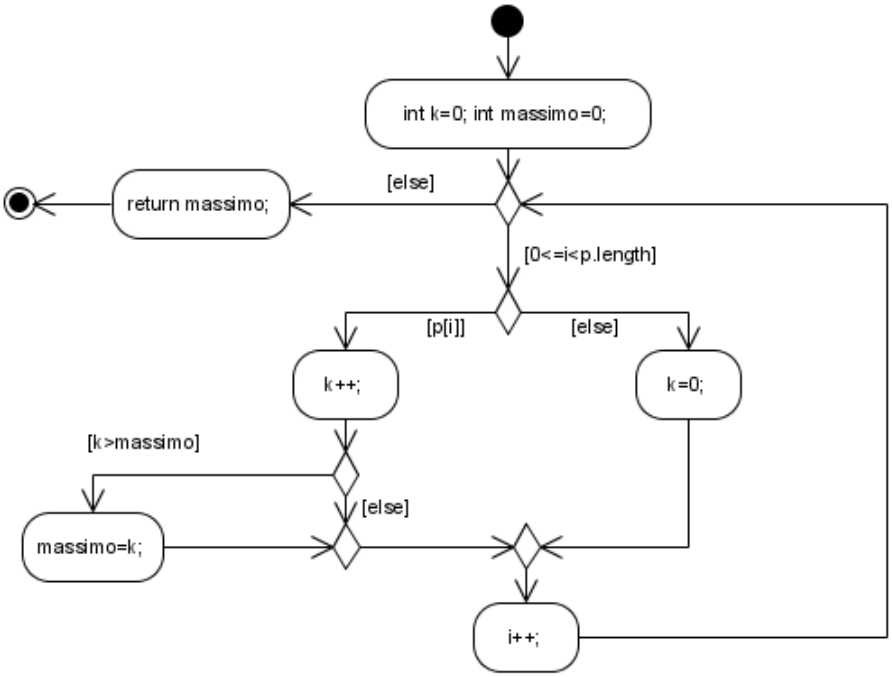
4. Albergo dei Fiori

Il seguente metodo determina la durata del più lungo periodo di occupazione di una stanza in un periodo dato.

```
public int massimoPeriodo (boolean [] p) {
     int k = 0;
     int massimo = 0;
     for (int i = 0; i < p.length; i++) {
       if (p[i]) {
           k++;
           if (k > massimo) {
             massimo = k;
           }
       } else {
           k = 0;
       }
     }
     return massimo;
}
```

Domanda. (Verifica).
a) Disegnare il grafo di flusso (o grafo di controllo) del metodo.
b) Dare un insieme di cardinalità minima di casi di prova per la copertura delle decisioni.

Risposta. Una possibile soluzione è la seguente:



Un insieme minimale di casi di prova che soddisfa la copertura richiesta è il seguente:

input	output
T F T	1

5. Grande distribuzione Modifica dei prezzi.

Il seguente metodo è di fantasia e non realizza esattamente alcun caso d'uso (perché?). E' stato scritto per controllare la lista dei nuovi prezzi rispetto al vincolo di mantenere le variazioni entro il 20%. Le variabili cl e db denotano gli oggetti che gestiscono rispettivamente le interazioni con il cliente e con il data base.

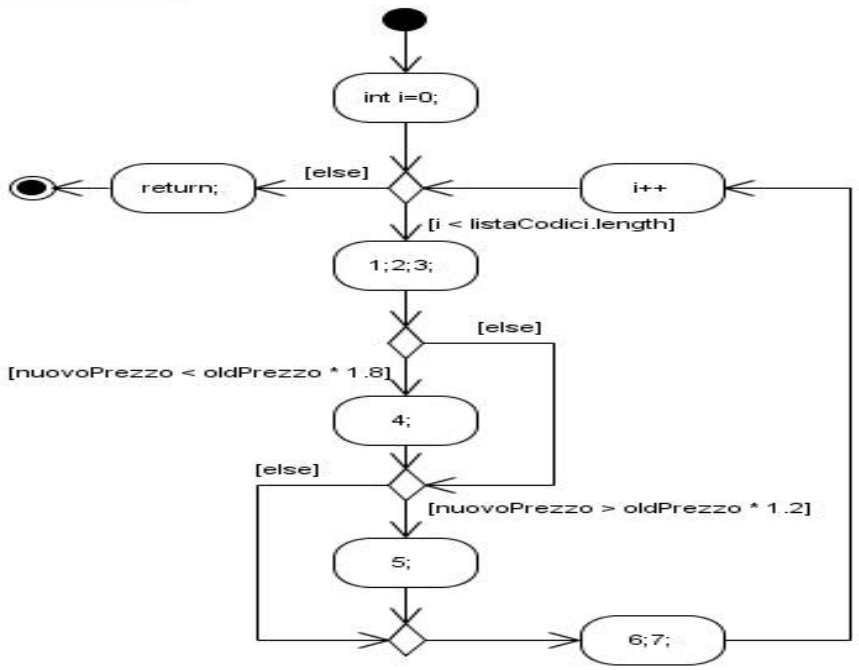
```
  public void controllaPrezzi(int [][] listaCodici) {
        for(int i=0; i < listaCodici.length; i++) {
            int codice = listaCodici[i][0];            //1
            int oldPrezzo = db.getPrezzo(codice);      //2
            int nuovoPrezzo = listaCodici[i][1];       //3
            if (nuovoPrezzo < oldPrezzo * 0.8){
               nuovoPrezzo = (int) (oldPrezzo * 0.8);  //4
            }
            if (nuovoPrezzo > oldPrezzo * 1.2) {
               nuovoPrezzo = (int) (oldPrezzo * 1.2);  //5
            }
            cl.aggiornato(codice,nuovoPrezzo);         //6
            db.setPrezzo(codice, nuovoPrezzo);         //7
        }
        return;
}
```

Dare un diagramma di attività con il grafo di flusso del metodo. Si usi la numerazione indicata per non ricopiare i comandi. Inoltre, per effettuare i test, si assuma che lo stub per getPrezzo sia definito in modo da restituire un valore pari a 10 volte il suo argomento. Gli argomenti dei metodi aggiornato e setPrezzo costituiscono l'output del metodo. Si utilizzi una tabella come la seguente per la risposta, con il numero di righe necessarie.

Input			**Output**	
i	**listaCodici[i]**			
	[0]	**[1]**	**codice**	**nuovoPrezzo**

- **a.** Dare un (solo!) caso di test che soddisfi il criterio (a scatola aperta) delle decisioni.
- **b.** Dare un insieme minimale di casi di test soddisfi il criterio (a scatola aperta) di copertura dei comandi.
- **c.** Dare un insieme minimale di casi di test soddisfi il criterio (a scatola aperta) di copertura dei cammini, nel caso di 1-test dei cicli.

Risposta: Il grafo di flusso:



a) Un possibile caso di test è il seguente

Input			Output	
i	listaCodici[i]			
	[0]	[1]	codice	nuovoPrezzo
0	1	11	1	11
1	2	15	2	16
2	3	40	3	36

b) Basta una lista con due prezzi, fuori dall'intervallo permesso, dalle due parti:

Input			Output	
i	listaCodici[i]			
	[0]	[1]	codice	nuovoPrezzo
1	2	15	2	16
2	3	40	3	36

c) Servono tre liste di un solo elemento, per fare tre cicli di un solo passo, in ciascuno dei quali si percorre uno dei tre cammini possibili nel corpo del ciclo:

Input			Output	
i	listaCodici[i]			
	[0]	[1]	codice	nuovoPrezzo
0	1	11	1	11

Input			Output	
i	listaCodici[i]			
	[0]	[1]	codice	nuovoPrezzo
0	2	15	2	16

Input			Output	
I	listaCodici[i]			
	[0]	[1]	codice	nuovoPrezzo
0	3	40	3	36

6. CicloPi

Per calcolare quanto deve essere pagato al momento dell'acquisto di un abbonamento, è stata definita la classe enumerazione TipoAbbonamento e scritto il codice seguente (per semplicità sono stati considerati solo due tipi di abbonamento).

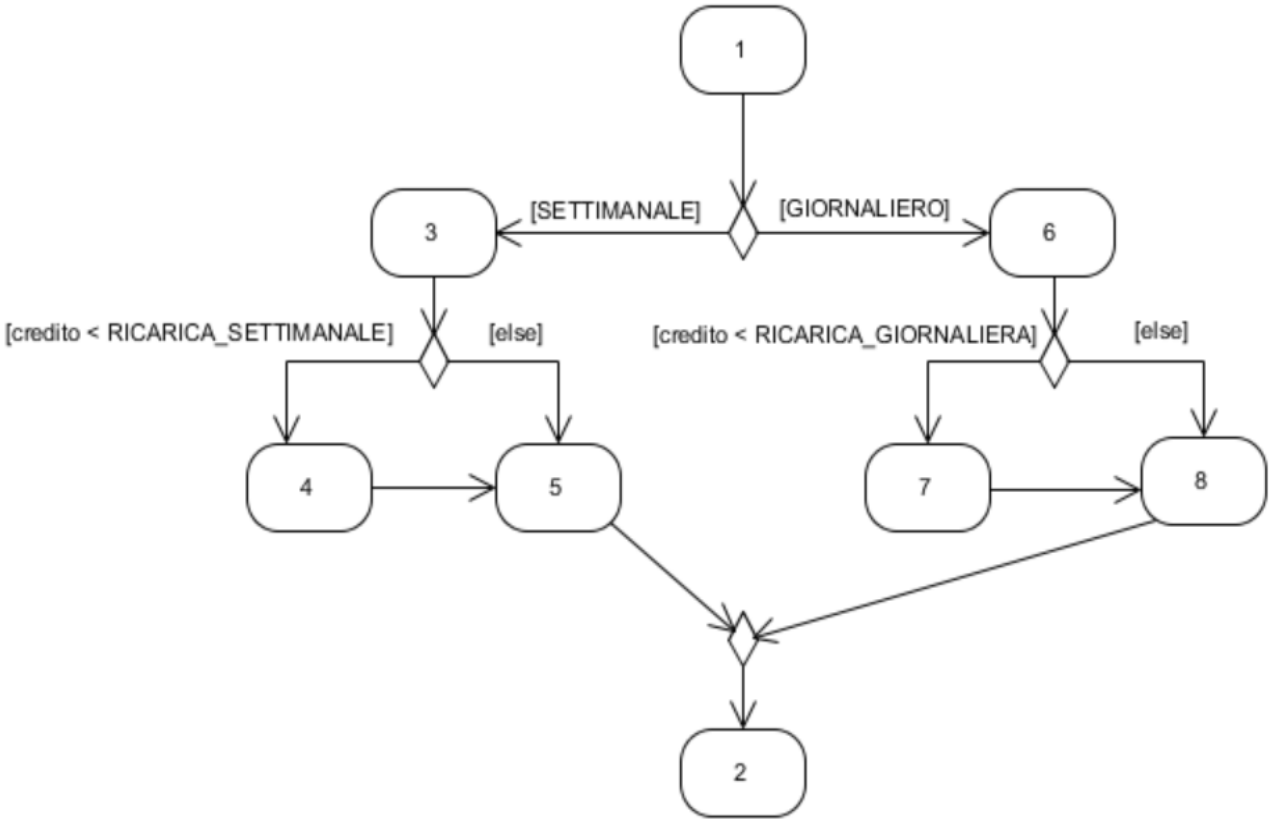
```
private final double COSTO_SETTIMANALE = 8.00;
private final double RICARICA_SETTIMANALE = 2.00;
private final double COSTO_GIORNALIERO = 4.00;
private final double RICARICA_SETTIMANALE = 1.00;
private double credito;

public double DRIVERcalcolaCosto(TipoAbbonamento abb, double credito){
   double costoAbbonamento = 0;
   this.credito = credito;                                          // 1

   switch(abb){
      case SETTIMANALE:
         costoAbbonamento = calcolaCostoSettimanale(); break;
      case GIORNALIERO:
         costoAbbonamento = calcolaCostoGiornaliero();
   }
   return costoAbbonamento;                                         // 2
}
```

```
private double calcolaCostoSettimanale( ) {
   double costo = COSTO_SETTIMANALE;                              // 3

   if (credito < RICARICA_SETTIMANALE)
     costo = costo + (RICARICA_SETTIMANALE – credito);            // 4

   return  costo;                                                 // 5
}

private double calcolaCostoGiornaliero( ) {
   double costo = COSTO_GIORNALIERO;                              // 6

   if (credito < RICARICA_GIORNALIERA)
     costo = costo + (RICARICA_GIORNALIERA – credito);            // 7

   return  costo;                                                 // 8
}
```

Domanda. Si disegni il grafo di flusso per il codice dato, etichettando i nodi, per semplicità, con i numeri associati alle linee di codice.

Si definisca un numero minimo di casi di test (senza considerare l'ambiente) per testare il metodo `DRIVERcalcolaCosto` (e i metodi che invoca) e avere 100% di copertura per il criterio dei cammini. Si giustifichi la risposta indicando i cammini usando le etichette dei nodi del grafo attraversati.

Risposta.



Input		Output	Giustificazione
Abb	credito		
SETTIMANALE	1.0	9.0	1,3,4,5,2
SETTIMANALE	3.0	8.0	1,3,5,2
GIORNALIERO	0.5	4.5	1,6,7,8,2
GIORNALIERO	2.0	4.0	1,6,8,2