Verifica: esercitazione

Laura Semini, Ingegneria del Software

Dipartimento di Informatica, Università di Pisa

Rebu1: euristica di selezione dell'autista

```
public Autista seleziona(Set<Autisti> disponibili, Richiesta r){
        int score=1000, d=0;
        Autista prescelto=null;
        Location aLoc=null, rLoc=r.getLocation();
        for (Autista a : disponibili) {
                aLoc=a.getLocation();    d=aLoc.distance(rLoc);
                if (d<score) {
                        score=d;
                        prescelto=a;  }
        }
        return a;
}
```

Si definiscano:

1. Un insieme minimo di valori di input che garantisca una copertura del 100% dei comandi;

2. Definire le proof obligations per avere il 100% di copertura delle decisioni e dare un rappresentante per classe;

3. Si svolga infine una ispezione strutturata del codice, definendo una checklist che includa la verifica del trattamento dei null. Si evidenzia qualche problema nel codice dato sopra?

Rebu1: RISPOSTA Definire un insieme minimo di valori di input che garantisca una copertura del 100% dei comandi.

```
public Autista seleziona(Set<Autisti> disponibili, Richiesta r){
1.    int score=1000, d=0;
2.    Autista prescelto=null;
3.    Location aLoc=null, rLoc=r.getLocation();
4.    for (Autista a : disponibili) {
5.          aLoc=a.getLocation();  d=aLoc.distance(rLoc);
6.          if (d<score) {
7.                score=d;
8.                prescelto=a;  }
      }
9.    return a;
}
```

Set<Autisti> disponibili	Richiesta r
{Andrea| Andrea.location=stazione }	r1 | r1.location= piazzaVittorio

Rebu1: RISPOSTA Definire le proof obligations per avere il 100% di copertura delle decisioni e dare un rappresentante per classe

```
public Autista seleziona(Set<Autisti> disponibili, Richiesta r){
      int score=1000, d=0;
      Autista prescelto=null;
      Location aLoc=null, rLoc=r.getLocation();
1.    for (Autista a : disponibili) {
2.          aLoc=a.getLocation();  d=aLoc.distance(rLoc);
3.          if (d<score) {
4.                score=d;
5.                prescelto=a;   }    }
      return a;}
```

Set<Autisti> disponibili	Richiesta r	Decisioni coperte (etichettate con le linee)
{ }	r1 | r1.location= p.zaVittorio	1 → esce subito dal ciclo
{Andrea| Andrea.location=stazione}	r1 | r1.location= p.zaVittorio	1→ 2,3 →4,5 ,1 →esce
{Andrea, Luigi | Andrea.location=stazione Luigi.location=torrePisa}	r1 | r1.location= p.zaVittorio	1→ 2,3 →4,5 , 1→2,3 → 1 → esce

Le soluzione chiedeva le proof obligations: { }, {un autista solo oppure n autisti ordinati dal più lontano al più vicino}, {due o più autisti ordinati dal più vicino al più lontano}

Rebu1: RISPOSTA Si svolga infine una ispezione strutturata del codice, definendo una checklist che includa la verifica del trattamento dei null. Si evidenzia qualche problema nel codice dato sopra?

```
public Autista seleziona(Set<Autisti> disponibili, Richiesta r){
      int score=1000, d=0;
      Autista prescelto=null;
      Location aLoc=null, rLoc=r.getLocation();
      for (Autista a : disponibili) {
            aLoc=a.getLocation();  d=aLoc.distance(rLoc);
            if (d<score) {
                  score=d;
                  prescelto=a;  }
      }
      return a;
}
```

- Un check è controllare il valore che viene assegnato alla variabile restituita→ difetto, variabile locale al ciclo for (se ne sarebbe comunque accorto il compilatore)
- Anche fosse stato `return prescelto` la variabile sarebbe potuta essere `null` (se `disponibili` è vuoto o se tutte le distanze calcolate sono >1000)

Rebu 2: emissioni

Le auto di REBU, circolando per molte ore nei centri cittadini, devono superare ogni anno un rigido test sui valori delle emissioni degli ossidi di azoto (NOx). Il metodo:

```
public void calcolaGiriMotore (double valoreSensoreAcceleratore)
```

dato un valore ricevuto dall'acceleratore dell'auto, calcola il numero di giri del motore, salva il risultato in un file di log e ordina al motore di girare a quel numero di giri. La centralina delle automobili implementa il metodo e lo invoca ogni decimo di secondo leggendo da un sensore sull'acceleratore.

In officina, la strumentazione di misura delle emissioni viene collegata via cavo alla centralina. Le emissioni vanno lette per tre soglie date di numero di giri. Per ogni soglia s_i, il meccanico accelera fino a quando la strumentazione di misura dice "ok" perché legge s_i dal file di log. A questo punto la strumentazione di misura raccoglie il gas di scarico per le analisi.

Rebu 2, emissioni: Si consideri la seguente implementazione

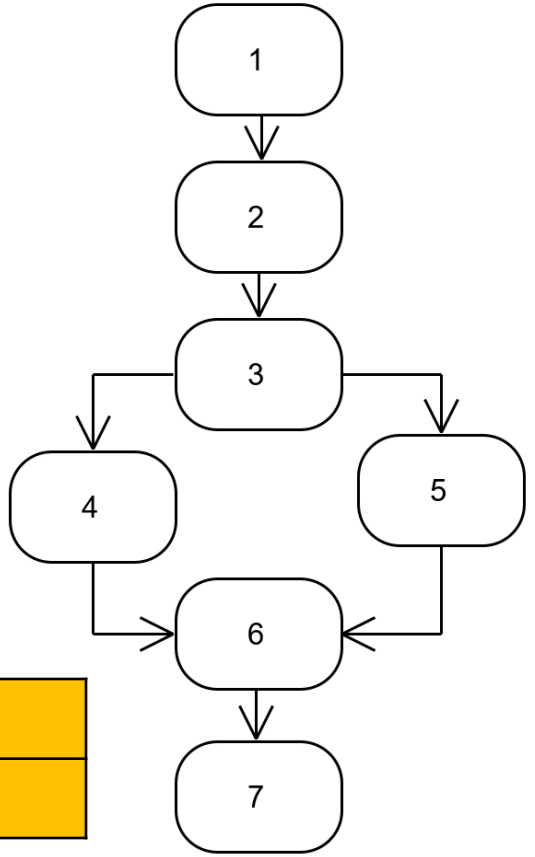
```
public void calcolaGiriMotore (double valoreSensoreAcceleratore){
      int numeroGiri, numeroGiriFake;
      \\ per una opportuna funzione f
      numeroGiri = f(valoreSensoreAcceleratore);
      if(inMovimento(ruoteMotrici)&!inMovimento(ruoteNonMotrici))
            numeroGiriFake = (int) 0.7 * numeroGiri;
      else numeroGiriFake = numeroGiri;
      … // scrivi numeroGiri nel file di log;
      … // invia numeroGiriFake al motore;
}
```

1. Si dia una definizione di difetto latente alla luce di questo esempio;
2. Applicando un criterio a scatola aperta, disegnare il grafo di flusso del metodo e dare un insieme minimo di valori restituiti dallo stub che realizza il metodo inMovimento(), per avere copertura al 100% delle decisioni;

Rebu 2, emissioni: RISPOSTA grafo di flusso del metodo e dare un insieme minimo di valori restituiti dallo stub che realizza il metodo inMovimento(), per avere copertura al 100% delle decisioni;

```
public void calcolaGiriMotore (double valoreSensoreAcceleratore){
1.      int numeroGiri, numeroGiriFake;
        \\ per una opportuna funzione f
2.      numeroGiri = f(valoreSensoreAcceleratore);
3.      if(inMovimento(ruoteMotrici)&!inMovimento(ruoteNonMotrici))
4.              numeroGiriFake = (int) 0.7 * numeroGiri;
5.      else numeroGiriFake = numeroGiri;
6.      … // scrivi numeroGiri nel file di log;
7.      … // invia numeroGiriFake al motore;
}
```

Si noti intanto che il valore del paramentro di input è completamente ininfluente per la copertura delle decisioni



1.	inMovimento(ruoteMotrici) = true	inMovimento(ruoteNonMotrici) = true
2.	inMovimento(ruoteMotrici) = true	inMovimento(ruoteNonMotrici) = false

Rebu 3

Si esegue un test del metodo calcolaSpesaSettimanale, che, dato un array di viaggi effettuati da un profilo business in una settimana, calcola la spesa totale. Si provano i seguenti casi di test, con il risultato riportato accanto a ciascuno. Indichiamo i viaggi con la notazione (*viaggio*, *costo*)

```
< [ ],0, _> risultato esecuzione: 0
< [(v1,5)], 5, _>  risultato esecuzione: 0
< [(v1,5),(v2,16),(v3,22)], 43, _> risultato esecuzione 38
```

Si riesce, da questi risultati, a ipotizzare eventuali difetti nel codice?

Si definisca un elemento di checklist per cercare altre occorrenze di difetti analoghi che possono essere presenti nel codice.

Rebu 3 RiSPOSTA

Si esegue un test black box del metodo calcolaSpesaSettimanale, che, dato un array di viaggi effettuati da un profilo business in una settimana, calcola la spesa totale. Si provano i seguenti casi di test, con il risultato riportato accanto a ciascuno. Indichiamo i viaggi con la notazione (…, *costo*), in cui i puntini astraggono dettagli non significativi.

```
< [ ],0, _> risultato esecuzione: 0

< [(v1,5)], 5, _>  risultato esecuzione: 0

< [(v1,5),(v2,16),(v3,22)], 43, _> risultato esecuzione 38
```

Si riesce, da questi risultati, a ipotizzare eventuali difetti nel codice?

→ **non viene considerato il primo elemento dell'array**

Si definisca un elemento di checklist per cercare altre occorrenze di difetti analoghi che possono essere presenti nel codice.

→ **difetti di tipo off-by-one/limiti degli array**

Rebu4: calcolaPartenzaPerAeroporto

Il metodo calcolaPartenzaPerAeroporto calcola l'orario di inizio di una corsa, conoscendo, l'anticipo, il tempo di percorrenza (che si assume qui indipendente dall'orario del volo), e il volo. Inoltre, restituisce un nuovo orario di partenza in caso di scostamenti superiori ai 15 minuti rispetto a un orario precedentemente calcolato.

Si consideri il seguente frammento di codice, dove il valore del parametro orarioPrevistoCorsa è -1 quando non è ancora stato calcolato un orario di partenza, diverso altrimenti:

(next slide)

Rebu4: calcolaPartenzaPerAeroporto

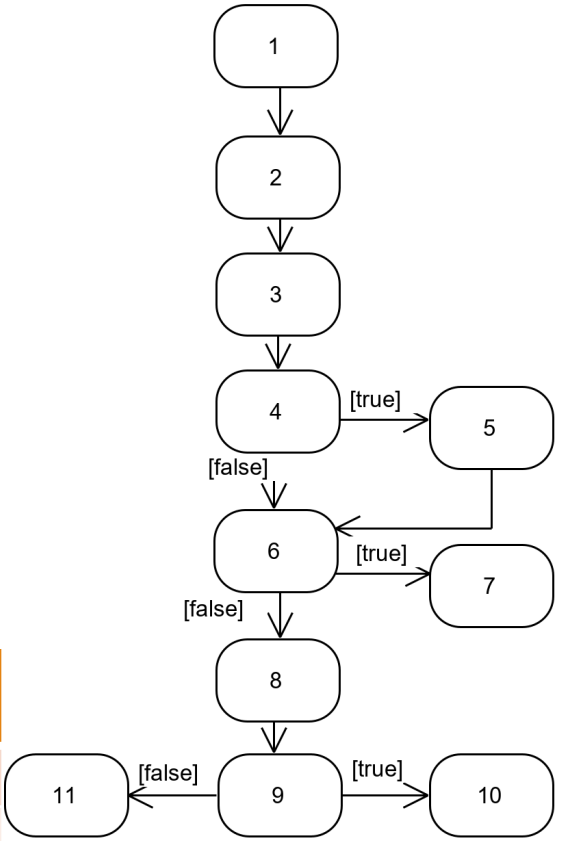
```
private int calcolaPartenzaPerAeroporto(int orarioPrevistoCorsa, int anticipo, int tempoPrecorrenza, Volo v){
        int oraPartenzaVolo = v.getOrarioPartenza();
        int new_orarioPrevistoCorsa = oraPartenzaVolo - anticipo - tempoPercorrenza ;
        int tempoAllaCorsa = new_orarioPrevistoCorsa - now();

        if (tempoAllaCorsa > 120 || orarioPrevistoCorsa == -1)   return new_orarioPrevistoCorsa ;
        if (tempoAllaCorsa <0)                                                  return -5;
        int differenza = new_orarioPrevistoCorsa – orarioPrevistoCorsa;
        if (Math.abs(differenza)>15)                    return new_orarioPrevistoCorsa ;
          else                                          return orarioPrevistoCorsa;
}
```

Dare il diagramma di flusso del metodo e un insieme minimo di casi di test per avere copertura (al 100%) delle decisioni e copertura delle condizioni (multiple condition coverage).

Si assuma che il tempo sia codificato in minuti a partire dalla mezzanotte, per esempio le 6:20 → 380

Rebu4: RISPOSTA Dare il diagramma di flusso del metodo e un insieme minimo di casi di test per avere copertura delle decisioni (homework: completare)

```
private int calcolaPartenzaPerAeroporto(int orarioPrevistoCorsa, int anticipo, int tempoPrecorrenza, Volo v){
1.      int oraPartenzaVolo = v.getOrarioPartenza();
2.      int new_orarioPrevistoCorsa = oraPartenzaVolo - anticipo - tempoPercorrenza ;
3.      int tempoAllaCorsa = new_orarioPrevistoCorsa - now();
4.      if (tempoAllaCorsa > 120 || orarioPrevistoCorsa == -1)
5.              return new_orarioPrevistoCorsa ;
6.      if (tempoAllaCorsa <0)
7.              return -5;
8.      int differenza = new_orarioPrevistoCorsa – orarioPrevistoCorsa;
9.      if (Math.abs(differenza)>15)
10.             return new_orarioPrevistoCorsa ;
11.     else    return orarioPrevistoCorsa;
}
```



orarioPrevistoCorsa	anticipo	tempoPrecorrenza	v.orarioP	now	archi
320	30	10	400	100	4->5
320	30	10	400	300	4->6; 6->8; 9->10
…	…	…..	…..	….	……
….					

Rebu4: RISPOSTA Dare il diagramma di flusso del metodo e un insieme minimo di casi di test per avere copertura copertura delle condizioni (multiple condition coverage) (oltre a quelli visti) (homework: completare)

```
private int calcolaPartenzaPerAeroporto(int orarioPrevistoCorsa, int anticipo, int tempoPrecorrenza, Volo v){
1.      int oraPartenzaVolo = v.getOrarioPartenza();
2.      int new_orarioPrevistoCorsa = oraPartenzaVolo - anticipo - tempoPercorrenza ;
3.      int tempoAllaCorsa = new_orarioPrevistoCorsa - now();
4.      if (tempoAllaCorsa > 120 || orarioPrevistoCorsa == -1)
5.              return new_orarioPrevistoCorsa ;
6.      if (tempoAllaCorsa <0)
7.              return -5;
8.      int differenza = new_orarioPrevistoCorsa – orarioPrevistoCorsa
9.      if (Math.abs(differenza)>15)
10.             return new_orarioPrevistoCorsa ;
11.     else    return orarioPrevistoCorsa;
}
```

I due casi di test della slide precedente coprono le decisioni. Per la multiple condition coverage aggiungere un caso di test che contenga tempoAllaCorsa<=120 e orarioPrevistoCorsa==-1

orarioPrevisto Corsa	anticipo	Tempo Precorrenza	v.orarioP	now	CONDIZIONI
-1	30	10	400	100	tempoAllaCorsa>120 true e orarioPrevistoCorsa ==-1
320	30	10	400	300	tempoAllaCorsa<=120 true

Rebu5

Si consideri il seguente frammento di codice, che ha lo scopo di controllare se un intervallo temporale A sia interamente contenuto all'interno di un intervallo temporale B (usato dal sistema REBU per controllare se una richiesta di disponibilità di auto condivisibile è compatibile con l'orario di disponibilità di una particolare auto messa in condivisione):

```
public boolean inside(Timespan a, Timespan b) {
        if (a.start < b.start)     return false;
        if (a.end > b.end)         return false;
        if ((a.start >= b.start) && (a.end <= b.end)) return true;
        return false;
}
```

Si adotti un atteggiamento di *defensive programming*, ovvero ci si proponga di realizzare una suite di test che consenta di verificare il funzionamento del metodo inside() senza fare assunzioni sulla correttezza dei parametri.

Oltre a dare la lista di casi di test, si commenti su quale criterio o insieme di criteri sono stati usati per generarla, e si propongano eventuali correzioni al codice (derivanti dai risultati del test).

Rebu5 SOLUZIONE

Si consideri il seguente frammento di codice, che ha lo scopo di controllare se un intervallo temporale A sia interamente contenuto all'interno di un intervallo temporale B (:

```
public boolean inside(Timespan a, Timespan b) {
                    if (a.start < b.start)  return false;
                    if (a.end > b.end)      return false;
                    if ((a.start >= b.start) && (a.end <= b.end))          return true;
                    return false;
}
```

Si adotti un atteggiamento di *defensive programming*, ovvero ci si proponga di realizzare una suite di test che consenta di verificare il funzionamento del metodo inside() senza fare assunzioni sulla correttezza dei parametri. Oltre a dare la lista di casi di test, si commenti su quale criterio o insieme di criteri sono stati usati per generarla, e si propongano eventuali correzioni al codice (derivanti dai risultati del test).

(a.start >a.end, lo stesso per b) cambiare il codice per fare un controllo

Test combinatorio

Si consideri la seguente proof obligation (3 parametri, 2 classi per i primi due, 3 classi per il terzo):

C1

- V1
- V2

C2

- V3
- V4

C3

- V5
- V6
- V7

Volendo fare un test esaustivo con tutte le combinazioni, quale sarebbe la dimensione della test suite ?

Test combinatorio RISPOSTA

Si consideri la seguente proof obligation:

C1

- V1
- V2

C2

- V3
- V4

C3

- V5
- V6
- V7

Volendo fare un test esaustivo con tutte le combinazioni, quale sarebbe la dimensione della test suite

Risposta: 2x2x3=12

Test combinatorio (Pairwise)

Si consideri la seguente proof obligation:

C1

- V1
- V2

C2

- V3
- V4

C3

- V5
- V6
- V7

Volendo considerare le sole combinazioni a coppie, quale sarebbe la test suite?

Test combinatorio (Pairwise) RISPOSTA

Si consideri la seguente proof obligation :

C1

- V1
- V2

C2

- V3
- V4

C3

- V5
- V6
- V7

Volendo considerare le sole combinazioni a coppie, quale sarebbe la test suite?→

Risposta: (6 casi di test)

V5	**V1**	**V4**
V5	**V2**	**V3**
V6	**V1**	**V3**
V6	**V2**	**V4**
V7	**V1**	**V3**
V7	**V2**	**V4**

Test combinatorio (if property)

Si ora consideri la seguente proof obligation :

C1

- V1 [property P1]
- V2 [property P2]

C2

- V3 [property P3]
- V4 [property P4]

C3

- V5 [if P1]
- V6 [if P4]
- V7

E ora, quale sarebbe la test suite?

Test combinatorio (if property) RISPOSTA

Si ora consideri la seguente specifica:

C1

V1 [property P1]
V2 [property P2]

C2

V3 [property P3]
V4 [property P4]

C3

V5 [if P1]
V6 [if P4]
V7

E ora, quale sarebbe la test suite?

Risposta: (8 casi di test)

V5	**V1**	**V3**
V5	**V1**	**V4**
V6	**V1**	**V4**
V6	**V2**	**V4**
V7	**V1**	**V3**
V7	**V2**	**V4**
V7	**V1**	**V4**
V7	**V2**	**V3**

Test combinatorio (if property e error)

Si ora consideri la seguente specifica:

C1

- V1 [property P1]
- V2 [property P2]

C2

- V3 [property P3]
- V4 [property P4]

C3

- V5 [if P1]
- V6 [if P4]
- V7 **[error]**

E ora, quale sarebbe la test suite?

Test combinatorio (if property e error) RISPOSTA

Si ora consideri la seguente specifica:

C1

- V1 [property P1]
- V2 [property P2]

C2

- V3 [property P3]
- V4 [property P4]

C3

- V5 [if P1]
- V6 [if P4]
- V7 **[error]**

E ora, quale sarebbe la test suite?

Risposta: (5 casi di test)

V5	**V1**	**V3**
V5	**V1**	**V4**
V6	**V1**	**V4**
V6	**V2**	**V4**
V7	**V1**	**V3**

Test mutazionale

```
 /* edit1( s1, s2 ) returns TRUE iff s1 can be
transformed to s2 by inserting, deleting, or
substituting a single character, or  by a no-op (i.e.,
if they are already equal).*/
…
6 int edit1( char *s1, char *s2) {
7 if (*s1 == '/0') {
8     if (*s2 == '/0?) return TRUE;
9 /* Only match is by deleting last char from
  s2 */
10    if ( *(s2 + 1) == '/0') return TRUE;
11    return FALSE;
12 }
13 if (*s2 == '/0') { /* Only match is by deleting
last char from s1 */
14    if (*(s1 + 1) == '/'0) return TRUE;
15    return FALSE;
16 }
17 /* Now we know that neither string is empty
*/
18 if (*s1 == *s2) {
19 return edit1(s1 +1, s2 +1);
20 }
21
22 /* Mismatch, here *s1 != *s2;
23 * only dist 1 possibilities are identical strings
after inserting, deleting, or substituting character
24 */
25
26 /* Substitution: We "look past" the
mismatched character */
27 if (strcmp(s1+1, s2+1) == 0) return TRUE;
28 /* Deletion: look past character in s1 */
29 if (strcmp(s1+1, s2) == 0) return TRUE;
30 /* Insertion: look past character in s2 */
31 if (strcmp(s1, s2+1) == 0) return TRUE;
32 return FALSE;
33 }
```

Si consideri la mutazione ottenuta sostituendo, alla linea 27, s1 +1 con s1 + 0.

Dare un caso di test che uccide il mutante

Definire:

Un mutante invalido;

Un mutante valido ma inutile (che fallisce per quasi ogni caso di test);

Un mutante valido ma equivalente;

Pisamover 3

Un Biglietto ha, tra i suoi attributi, l'ora di uscita prevista, in formato

LocalDateTime:

A date-time without a time-zone in the ISO-8601 calendar system, such as 2007-12-03T10:15:30.

Il metodo

int calcolaPrezzo(Biglietto b, LocalDateTime now)

di una qualche classe del vostro sistema restituisce il prezzo da pagare per saldare il parcheggio.

Definire (dandone il codice) un driver di test che sia anche factory di oggetti di tipo Biglietto per verificare la correttezza del metodo calcolaPrezzo(), garantendo una opportuna copertura secondo i criteri funzionali. Commentare le scelte fatte.

CicloPi: black box

CicloPi è gratuito per le corse di durata inferiore ai 30 minuti, anche più volte al giorno. Se l'utilizzo supera i 30 minuti consecutivi, sarà applicata la tariffazione relativa alla propria formula di abbonamento scalando l'importo dal credito presente sulla tessera. Il costo è di €0,90 la seconda mezz'ora (o frazione), €1,50 la terza, €2 dalla quarta mezz'ora in poi.

Rappresentando i valori della classe Data con gg/mm/aa – hh:mm, si è definito il metodo

double calcolaCostoBiciNonDanneggiata(Data dataInizio, Data dataFine)

calcola il costo di utilizzo di una bicicletta al momento della riconsegna.

Dare un insieme di casi di test progettati secondo i seguenti criteri a scatola chiusa: statistico, partizione dei dati di ingresso, frontiera.

CicloPi: risposta

oraInizio	oraFine	output	ragione
18/05/16-08:00	18/05/16-08:00	0.00	Caso limite, possibile per ripensamenti (o sella alta/bassa)
18/05/16-08:00	18/05/16-08:10	0.00	Partizione 1, statisticamente più probabile
18/05/16-08:00	18/05/16-08:15	0.00	Partizione 1, statisticamente più probabile
18/05/16-08:00	18/05/16-08:29	0.00	Partizione 1, statisticamente più probabile
18/05/16-08:00	18/05/16-08:30	0.90	Frontiera
18/05/16-08:00	18/05/16-08:45	0.90	Partizione 2
18/05/16-08:00	18/05/16-09:00	2.40	Frontiera
18/05/16-08:00	18/05/16-09:45	2.40	Partizione 3
18/05/16-08:00	18/05/16-09:30	4.40	Frontiera
18/05/16-08:00	18/05/16-09:45	4.40	Partizione 4
18/05/16-08:00	19/05/16-8:00	94:40	Frontiera 24 ore
18/05/16-08:00	19/05/16-8::15	94:40	Partizione 24 ore e rotti

Albergo dei Fiori

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

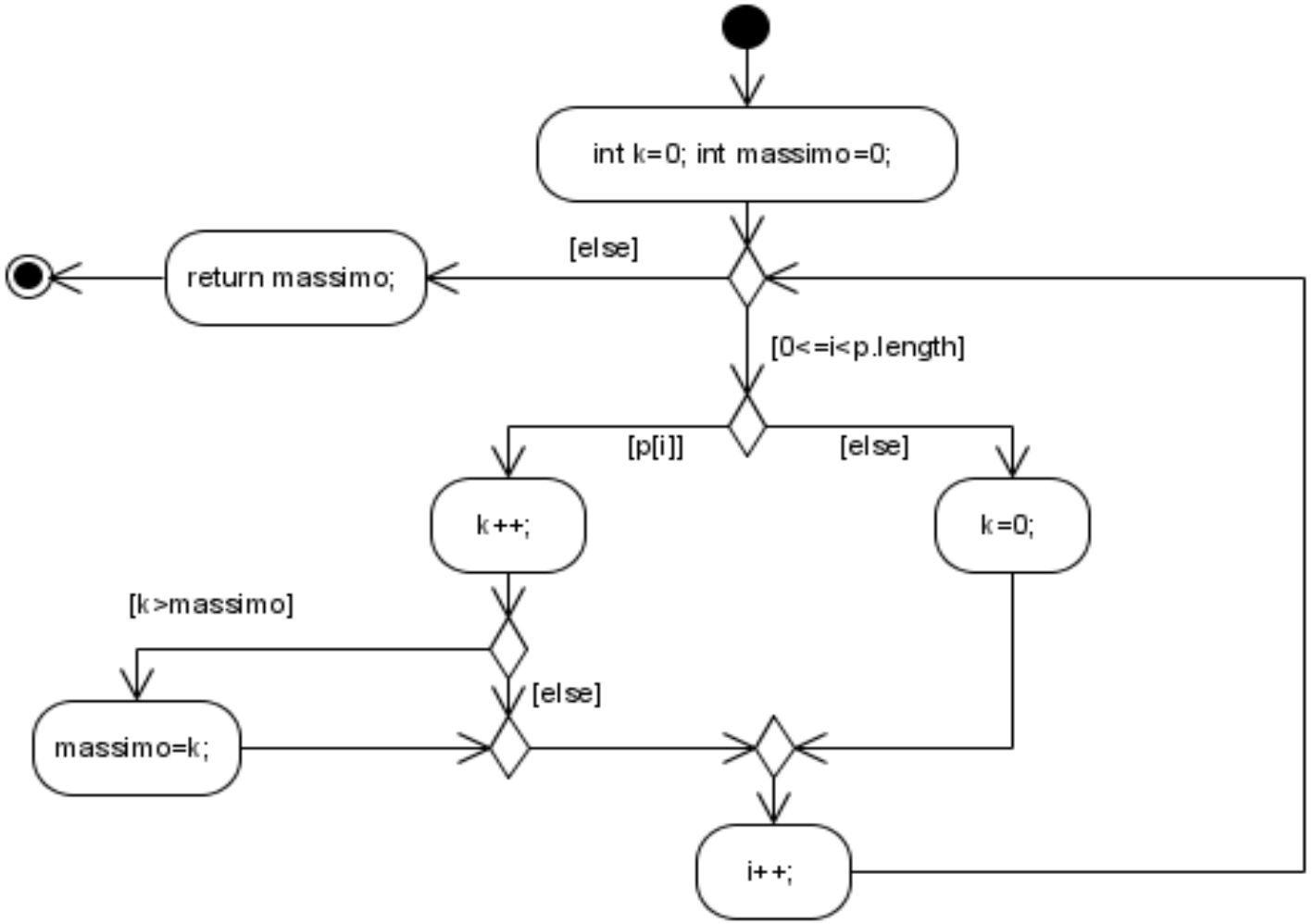
```
public int massimoPeriodo (boolean [] p) {
    int k = 0;
    int massimo = 0;
    for (int i = 0; i < p.length; i++) {
        if (p[i]) {
            k++;
            if (k > massimo) {
                massimo = k;
            }
        } else {
            k = 0;
        }
    }
    return massimo;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

Domanda.

a) Disegnare il grafo di flusso (o grafo di controllo) del metodo, usando un diagramma di attività

b) Dare un insieme di cardinalità minima di casi di prova per la copertura delle decisioni.

Albergo dei Fiori: risposta a)



Albergo dei Fiori: risposta b)

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

input	output
T F T	1

Homework: Si consideri il requisito: ogni settimana i giardinieri devono ricevere una email con la lista degli interventi manutentivi della settimana successiva

- Il metodo `assegna(Intervento[] i, Giardiniere[])`
- Riceve in input un array degli interventi e un array di giardinieri
- Restituisce un array di coppie (`Intervento, Giardiniere`) che distribuisce il lavoro in modo uniforme
- La classe Intervento implementa il metodo `int tempoStimato()` che restituisce il tempo stimato per il completamento dell'intervento.

Definire un ambiente e una batteria di test per testare il metodo assegna.

Si suggerisce di usare una factory per costruire gli oggetti di tipo intervento (o loro stub) nel driver