Compito Scritto di Ingegneria del Software

9 aprile 2010

Parte teorica, punti 14

Tempo a disposizione: 1 ora

Esercizio 4
Si descrivano le qualità esterne del prodotto software.

PUNTI 7

Esercizio 5
Si definiscano i principi di implementazione ed interfaccia di un modulo, descrivendone le relative prassi di buona progettazione.

PUNTI 7

Compito Scritto di Ingegneria del Software

9 aprile 2010

Parte pratica, punti 18

Tempo a disposizione: 1 ora e mezza

Esercizio n. 1

Modellare con una rete di Petri un sistema di controllo di un tornio. Il tornio può essere in tre stati: spento, rotante con velocità V1 e rotante con velocità V2. Il pannello dei comandi del tornio ha un pulsante che, quando premuto, fa passare il tornio fermo alla rotazione con velocità V1, fa passare il tornio in rotazione con velocità V1 alla rotazione con velocità V2 e ferma il tornio in rotazione con velocità V2.
Il tornio è dotato di un lampeggiante che è attivo solo quando il tornio è in movimento.
Inizialmente il tornio è fermo e il lampeggiante è spento.

PUNTI 5

Esercizio n. 2

Si dia una specifica in Z di un'officina meccanica. L'officina contiene n macchine utensili che possono fare lavorazioni di tipo 1 o 2 (non contemporaneamente). La lavorazione di un pezzo su una macchina richiede un'ora. Si supponga che siano disponibili 24 slot da un ora in una giornata per ogni macchina, numerati da 1 a 24. Si supponga che i pezzi siano numerati con un codice intero. Si modelli le operazioni di prenotazione e di esecuzione delle lavorazioni in una giornata.
Si modellino in Z le seguenti operazioni:
1) introduzione di un nuovo pezzo nel sistema: dato un codice, questo viene aggiunto all'elenco del pezzi dell'officina. L'operazione fallisce se il pezzo è già presente.
2) prenotazione di uno slot su una macchina: dato il codice di un pezzo, il tipo di lavorazione e l'ora, si aggiunga una prenotazione se esiste una macchina libera. Si restituisca il numero della macchina. Si ritorni errore se non ci sono macchine disponibili.
3) cancellazione di una prenotazione: dato il codice di un pezzo e il tipo di lavorazione, si rimuova la prenotazione corrispondente. Si ritorni errore se nessuna prenotazione per quel pezzo e per quella lavorazione è presente.
4) fine della lavorazione: data un'ora, si rimuovano tutte le prenotazioni in quell'ora su tutte le macchine.

PUNTI 7

Esercizio n.3

Si disegni il flusso di esecuzione e si esegua l'analisi di flusso, per mezzo di espressioni regolari, per le variabili del seguente programma.

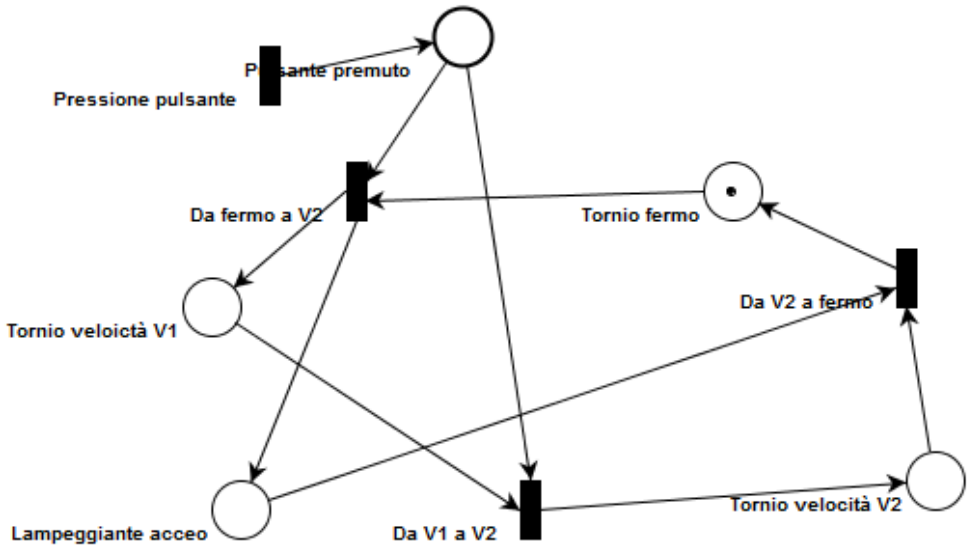
Si individuino inoltre eventuali sequenze non corrette di operazioni e, per ognuna, almeno un caso di test che porti alla sua esecuzione.

```
#include<stdio.h>
int main (void) {
     int a, b = 0, c;
     scanf("%d", &a);
     while (a > 0){
          a = a - b;
          b = b + 1;
          if (a <= 0)
               c = a / b;
     }
b = a - 1;
printf("%d\n", b + c * 2);
return 0;
}
```

PUNTI 6

Soluzione

Esericzio 1



Esercizio 2

Tipi definiti dall'utente:
[Ore, Macchine, Lavorazione]
Ore = 0..24
Macchine=1..n
Lavorazione=1..2
Variabili che descrivono lo stato del sistema:

1) prenotazioni è una funzione che associa a una macchina e a un'ora un codice di pezzo e una lavorazione;
2) pezzi è l'insieme dei pezzi

Officina

$$prenotazioni: Macchine \times Ore \pfun \mathbb{N} \times Lavorazione$$
$$pezzi: \mathbb{P}\ \mathbb{N}$$

$$ran\ prenotazioni \subseteq\ pezzi$$

Officina

ΔOfficina

---

pezzi’= $\varnothing$

prenotazioni’= $\varnothing$

Success

rep!: Report

---

rep! = 'Okay'

1) introduzione di un nuovo pezzo nel sistema

NuovoPezzo

ΔOfficina

pezzo?: $\mathbb{N}$

---

pezzo? $\notin$ pezzi

pezzi’= pezzi $\cup$ {pezzo?}

prenotazioni’= prenotazioni

PezzoGiàPresente

ΞOfficina

pezzo?: $\mathbb{N}$

rep!: Report

---

pezzo? $\in$ pezzi

rep! = ‘Pezzo già presente’

$$\text{Introduzione} \mathrel{\widehat{=}} \text{NuovoPezzo} \wedge \text{Success} \vee \text{PezzoGiàPresente}$$

2) prenotazione di uno slot su una macchina

PrenotazioneSlot

ΔOfficina
pezzo?: $\mathbb{N}$
lavorazione?: Lavorazioni
ora?: Ore
macchina!: Macchine

(macchina!,ora?) $\notin$ dom prenotazioni
prenotazioni' = prenotazioni $\cup$ {(macchina!,ora!) $\mapsto$ (pezzo?, lavorazione?)}

NessunaMacchinaLibera

ΞOfficina
pezzo?: $\mathbb{N}$
ora?: Ore

$\nexists$ macchina: Macchine $\bullet$ (macchina,ora?) $\notin$ dom prenotazioni
rep! = 'Nessuna macchina libera'

$$\text{VenditaSenzaCarta} \triangleq \text{PrenotazioneSlot} \wedge \text{Success} \vee \text{NessunaMacchinaLibera}$$

3) cancellazione di una prenotazione

Cancellazione

ΔOffcina
pezzo?: $\mathbb{N}$
lavorazione?: Lavorazioni
macchina?: Macchine
ora?: Ore

(macchina?,ora?) $\mapsto$ (pezzo?,lavorazione?) $\in$ prenotazioni
prenotazioni'= prenotazioni $\setminus$ {(macchina?,ora?) $\mapsto$ (pezzo?,lavorazione?) }

PrenotazioneNonPresente

ΞOffcina
pezzo?: $\mathbb{N}$
lavorazione?: Lavorazioni
macchina?: Macchine
ora?: Ore
rep!: Report

(macchina?,ora?) $\mapsto$ (pezzo?,lavorazione?) $\notin$ prenotazioni
rep! = 'Prenotazione non presente'

$$\text{CancellazionePrenotazione} \mathrel{\widehat=} \text{Cancellazione} \wedge \text{Success} \vee \text{PrenotazioneNonPresente}$$

4) fine della lavorazione: data un'ora, si rimuovano tutte le prenotazioni in quell'ora su tutte le macchine.

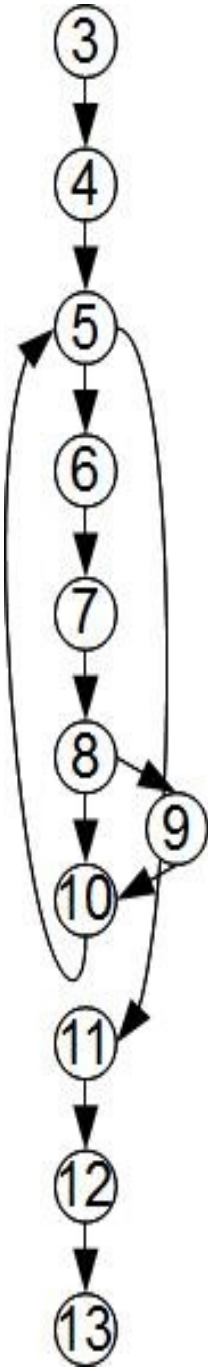
Fine

ΔOffcina
ora?: Ore

(macchina?,ora?) $\mapsto$ (pezzo?,lavorazione?) $\in$ prenotazioni
prenotazioni'= prenotazioni $\setminus$ {macchina: Macchine | (macchina,ora?) $\in$ dom prenotazioni $\bullet$ (macchina,ora?) $\mapsto$ prenotazioni(macchina,ora?) }

$$\text{FineLavorazione} \mathrel{\widehat=} \text{Fine} \wedge \text{Success}$$

```
1. #include<stdio.h>
2. int main (void) {
3. int a, b = 0, c;
4. scanf("%d", &a);
5. while (a > 0){
6. a = a - b;
7. b = b + 1;
8. if (a <= 0)
9. c = a / b;
10.     }
11.     b = a - 1;
12.     printf("%d\n", b + c * 2);
13.     return 0;
14.     }
```



	a	b	c
3	a	ad	a
4	d		
5	u		
6	ud	u	
7		ud	
8	u		
9	u	u	d
11	u	d	
12		u	u

Le espressioni regolari delle variabili a,b,c sono:

a: a d u(ud u(u+ε) u)* u
b: ad (u ud (u+ε))* d u
c: a ((d+ε))* u

La variabile c potrebbe essere usata senza essere definita. Questo accade in due casi:

1. Se il ciclo while non viene mai eseguito (condizione a>0 falsa), quindi per a<=0 inserita dall'utnte.

2. Se il ciclo while finisce senza che sia mai stata vera la condizione dell'if. Questo però non accade mai, perché la condizione di uscita del ciclo (che sia falso a>0, quindi a<=0) implica la verità della condizione dell'if.