PCAD 23/24 - Esercizi 2

1 Esercizio 1

Consideriamo il programma concorrente descritto qui sotto.

```
semaphore S=1,T=0;
```

```
Process P
S.wait()
print("P")
T.signal();
```

```
Process Q
T.wait()
write("Q")
S.signal()
```

Figura 1: Programma semplice con semafori

1. Quale le stampe possibile di questo programma?
2. Cosa cambia se cancelliamo l'istruzione `S.wait()`?
3. Cosa cambia se cancelliamo l'istruzione `T.wait()`?

2 Esercizio 2

Consideriamo l'algoritmo di Barz, descritto qui sotto, che permette a al massimo $k > 0$ processi di accedere alla sezione critica.

```
semaphore S= 1
semaphore gate = 1
int count = k
```

```
Process P
   while(true){
p1:  non-critical section
p2:  gate.wait()
p3:  S.wait()
p4:  count=count-1
p5:  if count > 0{
p6:    gate.signal()}
p7:  S.signal()
p8:  critical section
p9:  S.wait()
p10: count=count+1
p11: if count == 1{
p12:   gate.signal()}
p13: S.signal()
    }
```

Figura 2: Algoritmo di Barz

1. Spiegare perché possiamo riscrivere l'algoritmo di Barz nella forma semplificata proposta qui sotto.
2. Scriviamo $\#q3$ [rispettivamente $\#q4$] per rappresentare il numero di processi che sono in $q3$ [rispettivamente $q4$]. Dimostrare che la congiunzione delle formule successive è un invariante:
   (a) $q3 \Rightarrow (gate = 0)$
   (b) $q3 \Rightarrow (count > 0)$

(c) $\#q3 \leq 1$

(d) $((gate = 0) \wedge \neg q3) \Rightarrow (count = 0)$

(e) $(count \leq 0) \Rightarrow (gate = 0)$

(f) $(gate = 0) \vee (gate = 1)$

3. Dimostrare che la formula $count = k - \#q4t$ è un invariante.

4. Dedurre che ad ogni momento il numero di processi in sezione critica è inferiore o uguale a k.

```
semaphore S= 1
semaphore gate = 1
int count = k
```

```
Process P
   while(true){
q1:  non-critical section
q2:  gate.wait()
q3:  atomic{
       count=count-1
       if count > 0{
         gate.signal()}
     }
q4:  critical section
q5:  atomic{
       count=count+1
       if count == 1{
         gate.signal()}
     }
    }
```

Figura 3: Algoritmo di Barz semplificato

3 Esercizio 3

In una città tranquilla, un parrucchiere ha un piccolo salone con una porta d'ingresso, una porta di uscita, una poltrona da parrucchiere e N sedie. I clienti arrivano dalla porta principale ed escono dalla porta di uscita dopo il taglio dei loro capelli. Dato che il salone è piccolo, solo il cliente sulla poltrona da parrucchiere può essere servito dal parrucchiere in un determinato momento. Il parrucchiere passa la sua vita tra dormire e servire i suoi clienti. Quando non ha clienti, il parrucchiere dorme. Quando arriva un cliente e la poltrona è libera, si siede e sveglia il parrucchiere. Se la poltrona non è libera, il cliente occupa una sedia se ci sono sedie libere o attende che una sedia si renda disponibile altrimenti. Un cliente su una sedia attende che la poltrona sia disponibile. Dopo avere terminato il taglio, il parrucchiere manda fuori il cliente e va a dormire.

Proporre un modello per questo problema utilizzando i semafori per la sincronizzazione tra il parrucchiere e i suoi clienti. Per questo, bisogna scrivere:

- un processo `Cliente` con i stati: in attesa di una sedia, su una sedia in attesa della poltrona, sulla poltrona in attesa della fine del suo taglio e fuori;
- un processo `Parrucchiere` con gli stati inattivo e in servizio;
- le dichiarazioni dei semafori con le loro valori iniziali.