Esercizio E2.7

Parte a)

Impostazione

La soluzione è molto semplice: in pratica è come se due mailbox di uguale dimensione fossero sovrapposte una all'altra: quella da cui riceve `R1` e quella da cui riceve `R2`. A partire dalla classica soluzione realizzata mediante un buffer circolare di `N` dimensioni, è sufficiente duplicare i contatori degli elementi pieni del buffer, uno (`cont1`) per contare i messaggi che devono essere ricevuti da `R1` e l'altro (`cont2`) per `R2`. Analogamente saranno necessari due diversi puntatori agli elementi del buffer, uno per indicare a `R1` da dove prelevare il prossimo messaggio (`testa1`) e analogamente per `R2` (`testa2`). Infine, saranno necessarie due diverse variabili condition su cui sospendere rispettivamente `R1` re `R2` (`non_vuoto1` e `non_vuoto2`).

```
monitor mailbox {
   mes buffer[N];   /*buffer circolare*/
   int testa1=0, testa2; /*puntatori al buffer per R1 e R2*/
   int coda=0; /*puntatore al buffer per i mittenti*/
   int cont1=0, cont2=0; /*contatori degli elementi pieni del buffer per R1 e R2*/
   condition non_pieno, non_vuoto1, non_vuoto2;
```

/*il numero di elementi vuoti del `buffer` corrisponde alla differenza fra N e il massimo fra `cont1` e `cont2`. In particolare, il `buffer` non contiene elementi vuoti se (`cont1==N`) oppure se (`cont2==N`).
Inoltre, dopo la ricezione di un messaggio da parte di uno dei due riceventi, non viene liberato necessariamente l'elemento del `buffer` da cui il messaggio è stato estratto. Ciò accade solo se il messaggio è già stato estratto anche dall'altro ricevente. E cioè, se dopo l'estrazione, e una volta decrementato il contatore relativo al processo ricevente, il valore di quest'ultimo è ancora maggiore o uguale al contatore dell'altro ricevente*/

```
   public void send (mes x){
      if (cont1==N || cont2==N) wait(non_pieno); /*buffer pieno*/
      buffer[coda]=x;
      coda = (coda+1)%N;
      /*un nuovo messaggio è disponibile sia per R1 che per R2*/
      cont1++;
      cont2++;
      signal(non_vuoto1);
      signal(non_vuoto2);
   }

   public mes receive1(){
      mes y;
      if (cont1==0) wait(non_vuoto1); /*buffer vuoto per R1*/
      y = buffer[testa1];
      testa1 = (testa1+1)%5;
      cont1--;
      if(cont1 >= cont2)signal(non_pieno); /*un elemento del buffer è stato
                                             completamente liberato*/
      return y;
   }
```

/*la `receiv2` è del tutto identica alla `reeive1` scambiando 1 con 2 e viceversa*/

```
   public mes receive2(){
      mes y;
      if (cont2==0) wait(non_vuoto2);
      y = buffer[testa2];
      testa2 = (testa1+2)%5;
      cont2--;
      if(cont2 >= cont1)signal(non_pieno);
      return y;
   }

}
```

Parte b)

Impostazione

In questo caso, per garantire il rispetto dei vincoli di priorità possiamo imporre che ogni `receive1` sia preceduta da una `send` e che ogni `receive2` sia preceduta da una `receive1`.
Per questo, alla fine di una `send` viene incrementato solo il contatore `cont1` dei messaggi che possono essere ricevuti da `R1`, svegliando quest'ultimo se è in attesa di un messaggio. Analogamente, alla fine della `receive1` viene incrementato solo il contatore `cont2` dei messaggi che possono essere ricevuti da `R2`, svegliando quest'ultimo se è in attesa di un messaggio. Infine, il completamento di una receive2 implica la liberazione di un elemento del `buffer`. Con questo criterio, infatti, il numero di elementi liberi del `buffer` coincide con la differenza fra `N` e il numero di messaggi che `R2` deve ancora ricevere (`cont2`).

```
monitor mailbox {
   mes buffer[N];
   int testa1=0, testa2, coda=0; cont1=0, cont2=0;
   condition non_pieno, non_vuoto1, non_vuoto2;

   public void send (mes x){
      if (cont2==N) wait(non_pieno);
      buffer[coda]=x;
      coda = (coda+1)%N;
      cont1++;
      signal(non_vuoto1);
   }

   public mes receive1(){
      mes y;
      if (cont1==0) wait(non_vuoto1);
      y = buffer[testa1];
      testa1 = (testa1+1)%5;
      cont1--;
      cont2++;
      signal(non_vuoto2);
      return y;
   }

   public mes receive2(){
      mes y;
      if (cont2==0) wait(non_vuoto2);
      y = buffer[testa2];
      testa2 = (testa1+2)%5;
```

```
        cont2--;
        signal(non_pieno);
        return y;
    }

}
```