Esercizio E2.3

Impostazione

1. Quali processi?
   Esistono due tipi di processi, i clienti normali e quelli che chiedono la sostituzione di un articolo. Per distinguerli si introduce una variabile **t** definita per enumerazione che rappresenta i due tipi di richiesta

2. Quale struttura per i processi

```
cliente
   negozio.entra;
   <guarda cosa c'è di bello>
   negozio.inizia_attività(t);
   <acquista o cambia>
   negozio.fine_attività(t);
   <paga o passa>
   negozio.esci (t);
```

dove **negozio** rappresenta la particolare istanza del tipo di monitor **negozio_sportivo.**

SOLUZIONE

```
#define   C  … ;  /* numero di commessi */
#define   CAP … ;  /* capacità del negozio */
typedef enum {N, S}richiesta;   /* richiesta di un cliente Normale e per Sostituzione */

monitor negozio_sportivo {
   /* variabili del monitor: */
   int  commessi_occupati=0;  /* numero di commessi occupati */
   int  posti_liberi=CAP ;  /* posti liberi nel negozio */
   boolean super_occupato=false ;   /* stato del supervisore */
   boolean  cassa_occupata=false;   /*stato della cassa */
   condition coda_fuori;    /* coda di entrata*/
   condition coda_dentro [2] ;  /* clienti che aspettano di eseguire l'attività di acq/sost*/
   condition coda_cassa ;  /* coda alla cassa (clienti normali) */

/*operazioni del monitor (entry): */

   public entry void entra{
      if (posti_liberi==0)  /* controllo se c'è un posto libero */
         wait (coda_fuori);
      posti_liberi -- ;  /*sono entrato */
```

```
   }

   public entry void inizio_attività (richiesta t) {
      if (t==N) { /*cliente normale */
         if (commessi_occupati==C)
            wait ( coda_dentro[t]);
         commessi_occupati ++ ;
      }
      else { /*t==S */
         if ((commessi_occupati == C)||(super_occupato))
            wait (coda_dentro[t]);
         commessi_occupati ++ ;
         super_occupato = true;
      }
   }

   public entry void  fine_attività (richiesta t) {
      commessi_occupati -- ; /* il commesso viene liberato*/
      if (t==S)
         super_occupato = false; /* il supervisore e` libero*/
      /*risveglio un cliente dando la priorità a quelli normali: */
      if (!empty (coda_dentro[N]))
         signal(coda_dentro[N]);
      else if ((!empty(coda_dentro[S])) && (!super_occupato))
               signal(coda_dentro[S]);
      if (t = N) {  /* il cliente normale va alla cassa:*/
         if (cassa_occupata)    /*controllo se la cassa è libera */
               wait(coda_cassa);
         cassa_occupata = true;
      }
   }

   public entry void  esci (tipo t) {
      if (t==N) { /* il cliente normale libera la cassa*/
         cassa_occupata = false;
         if (!empty (coda_cassa))
            signal(coda_cassa);
      }
      posti_liberi ++ ; /* il posto viene liberato:*/
      if (!empty(coda_fuori))
         coda_fuori.signal; /*risveglio un solo cliente */
   } /* fine entry*/

} /* fine monitor */
```