Esercizio 5

In un programma multithread, ogni thread esegue il seguente codice:

```
void *thread(void *arg)
{
  int voto = rand()%2;

  vota(voto);

  if (voto == risultato()) printf("Ho vinto!\n");
  else printf("Ho perso!\n");

  pthread_exit(0);
}
```

cioe' ogni thread:

- esprime un voto, che puo' essere 0 o 1, invocando la funzione vota(), la quale registra il voto in una struttura dati condivisa che per comodita' chiameremo "urna";

- aspetta l'esito della votazione invocando la funzione risultato(), la quale controlla l'urna e ritorna 0 o 1 a seconda che ci sia una maggioranza di voti 0 oppure di voti 1.

- se l'esito della votazione e' uguale al proprio voto, stampa a video la stringa "Ho vinto", altrimenti stampa la stringa "Ho perso";

Supponiamo che ci siano un numero dispari di threads nel sistema. Il candidato deve implementare la struttura dati

```
struct {
  ...
} urna;
```

e le funzioni:

```
void vota(int v);

int risultato(void);
```

in modo che i thread si comportino come segue:

- Se l'esito della votazione non puo' ancora essere stabilito, la funzione risultato() deve bloccare il thread chiamante. Non appena l'esito e' "sicuro" (ovvero almeno la meta' piu' uno dei threads ha votato 0, oppure almeno la meta' piu' uno dei threads ha votato 1) il thread viene sbloccato e la funzione risultato() ritorna l'esito della votazione. I thread vengono sbloccati il piu' presto possibile, quindi anche prima che abbiano votato tutti.

Utilizzare i costrutti pthread_mutex_xxx e pthread_cond_xxx visti a lezione.