Algoritmi i strukture podataka

- predavanja -

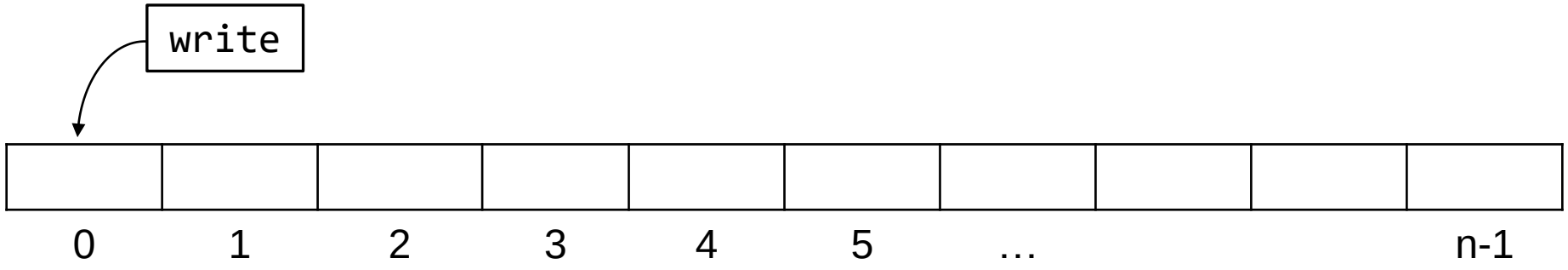
7. Red

Red

- struktura podataka kod koje se prvi pohranjeni podatak prvi uzima u obradu (First In First Out – FIFO)
- potrebne operacije:
  - dodavanje (*enqueue*) elemenata u red
  - brisanje (*dequeue*) elemenata iz reda
  - inicijalizacija praznog reda
  - po potrebi
    - uvid u sadržaj početka reda (*peek*)
- pojedina operacija *enqueue* ili *dequeue* zahtijeva jednako vremena bez obzira na broj pohranjenih podataka
- može se realizirati statičkom strukturom podataka

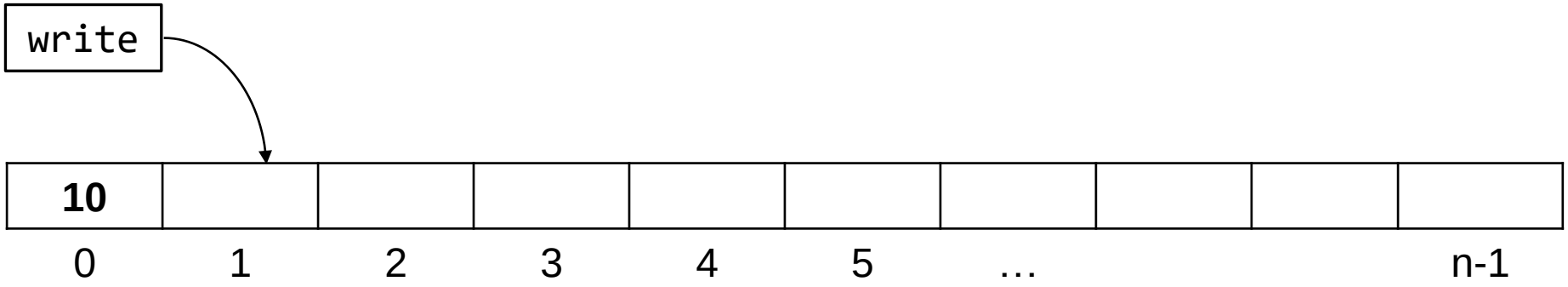
S jednom oznakom (za mjesto upisivanja)?

- Prazan red



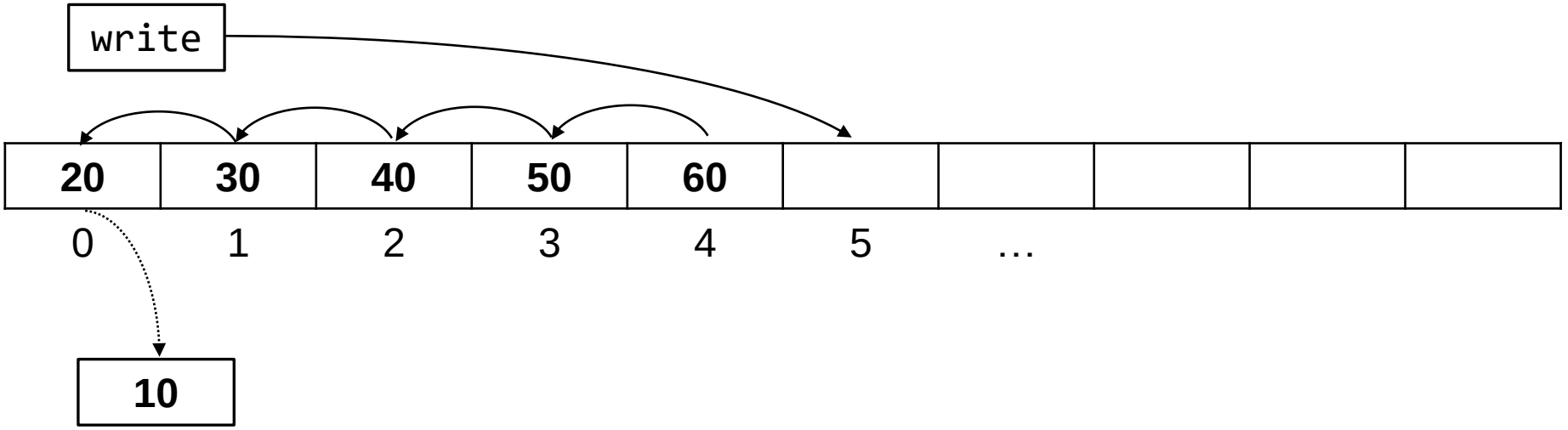
- *enqueue:*

```
if (write < n) queue[write++] = item;
```



S jednom oznakom (za mjesto upisivanja)?

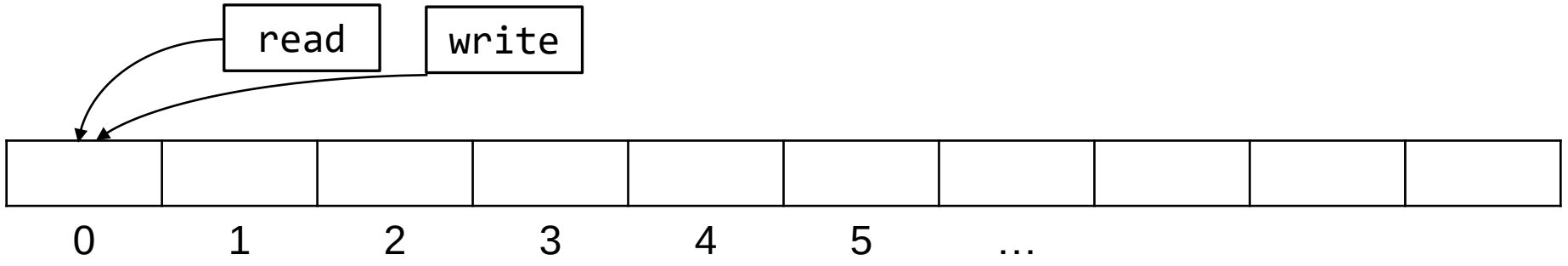
- Ali, *dequeue*



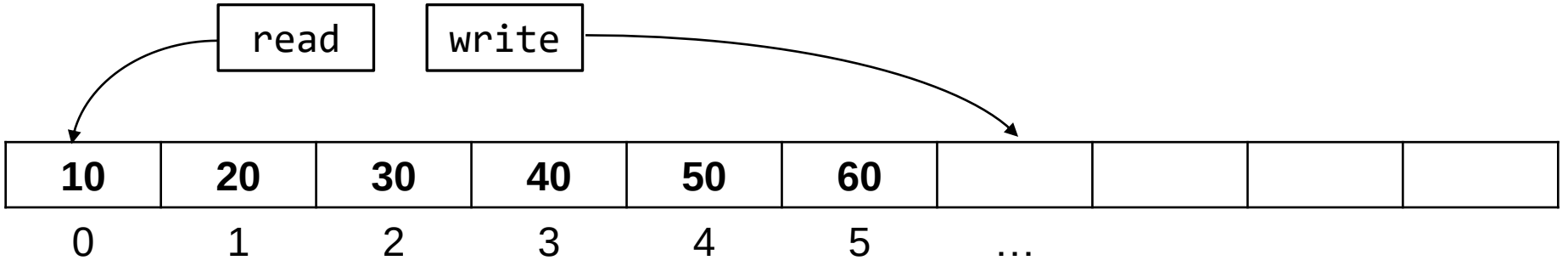
- O(n) !!!

S oznakama za mjesto čitanja i upisivanja

- Prazan red



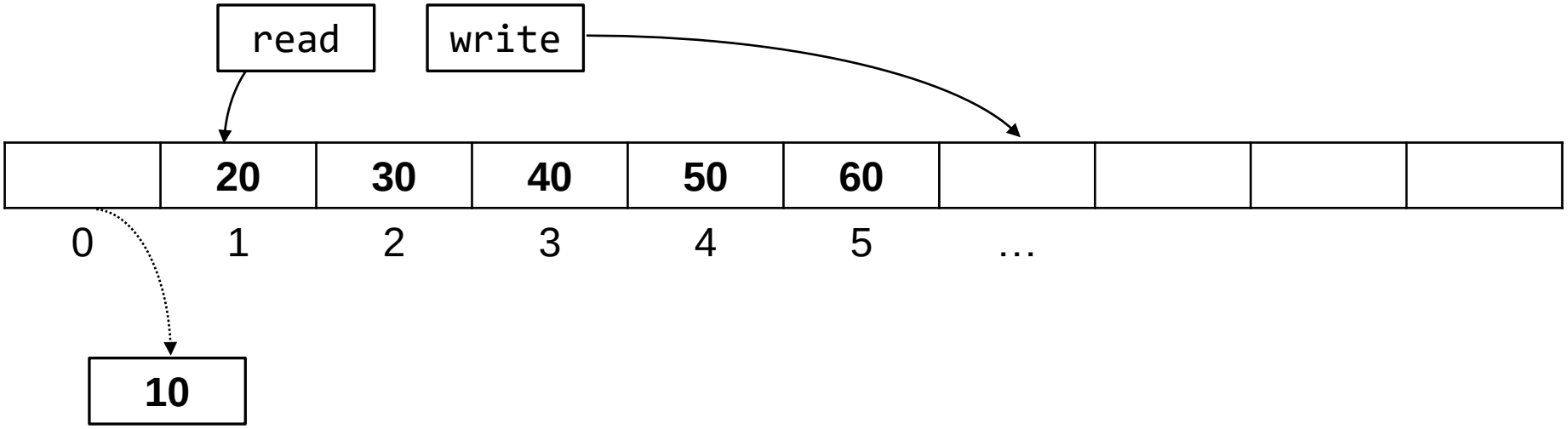
- Neprazan red



S oznakama za mjesto čitanja i upisivanja

- *dequeue*

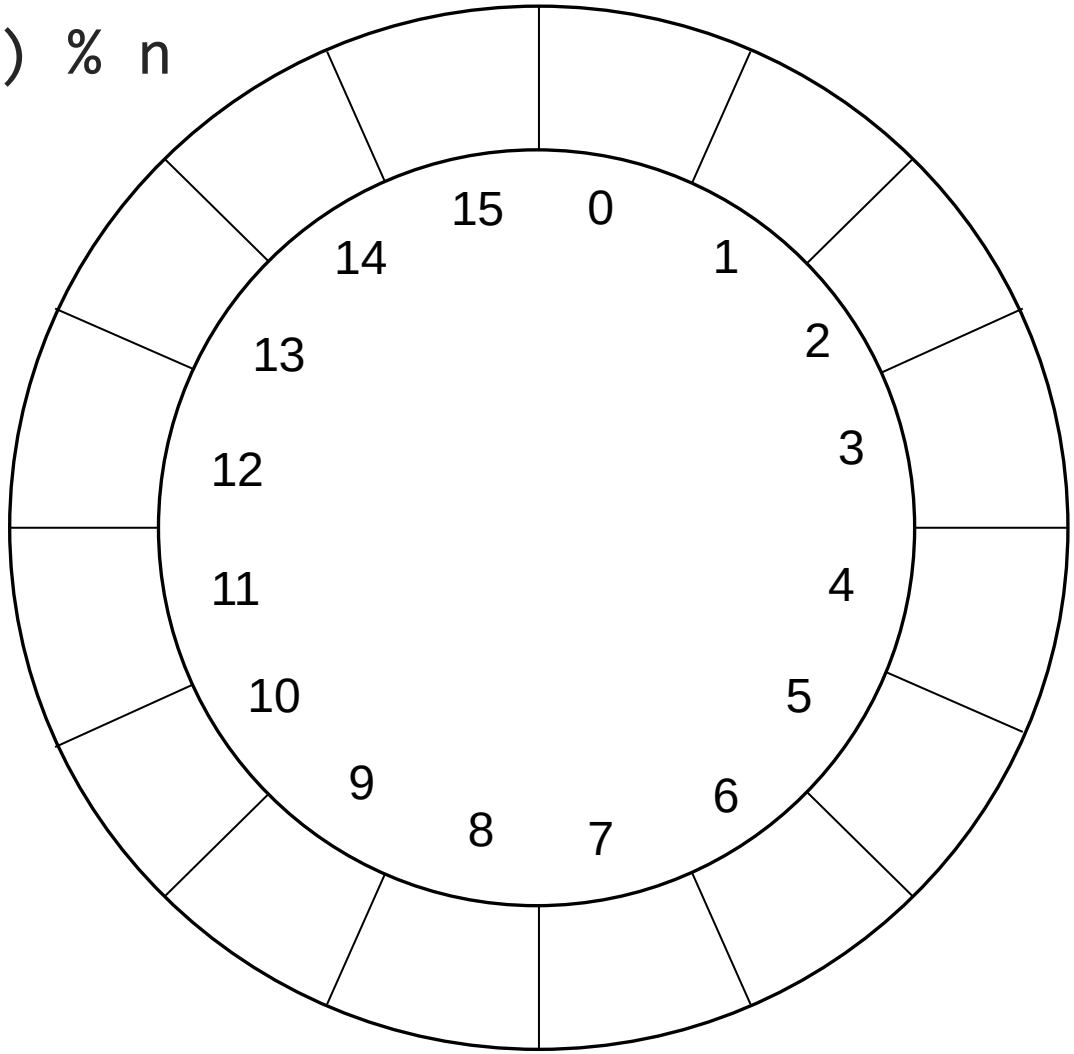
```
if (read < n) item = queue[read++];
```



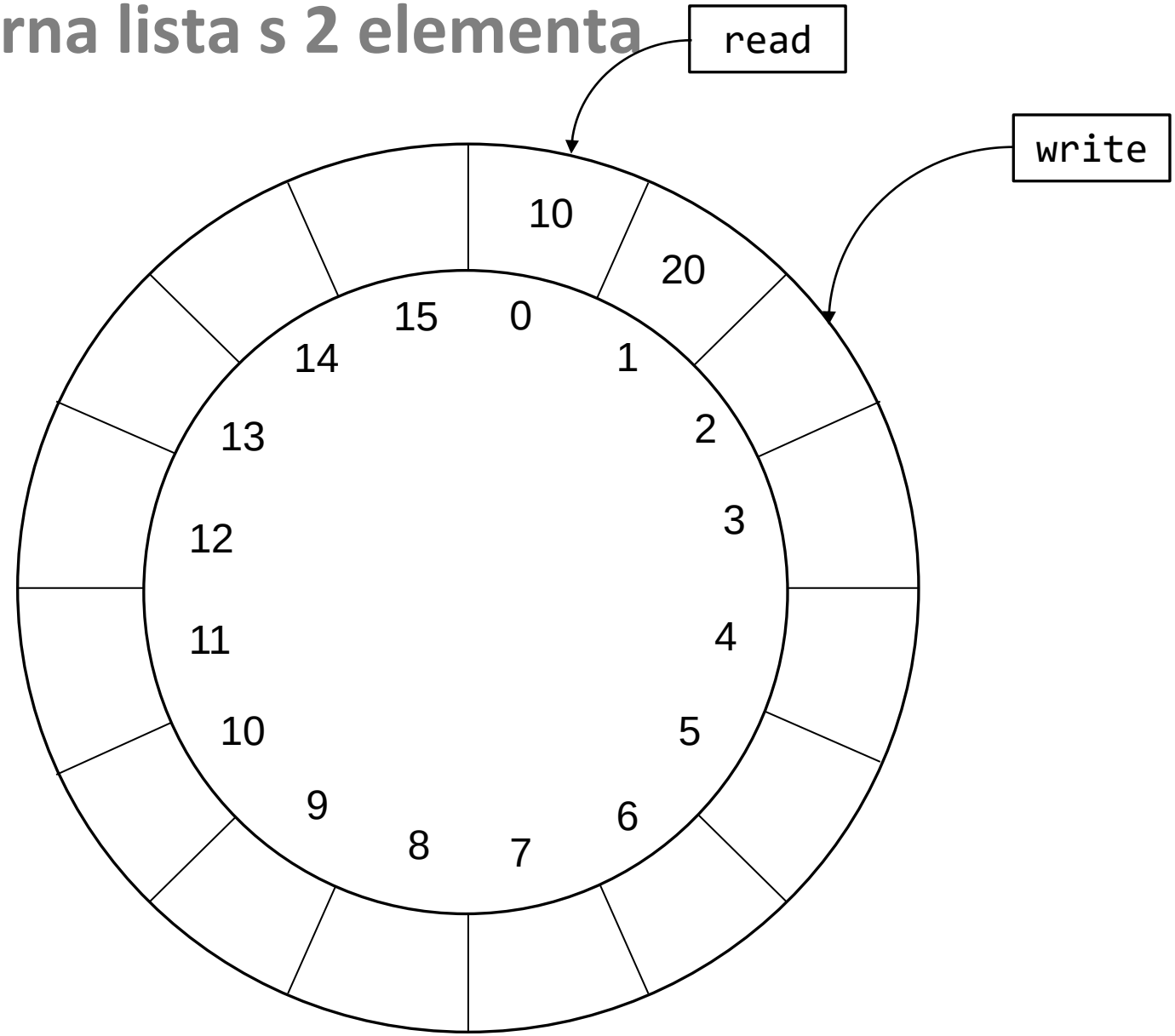
- što nakon više uzastopnih *queue* i *dequeue*?

Rješenje: cirkularnost

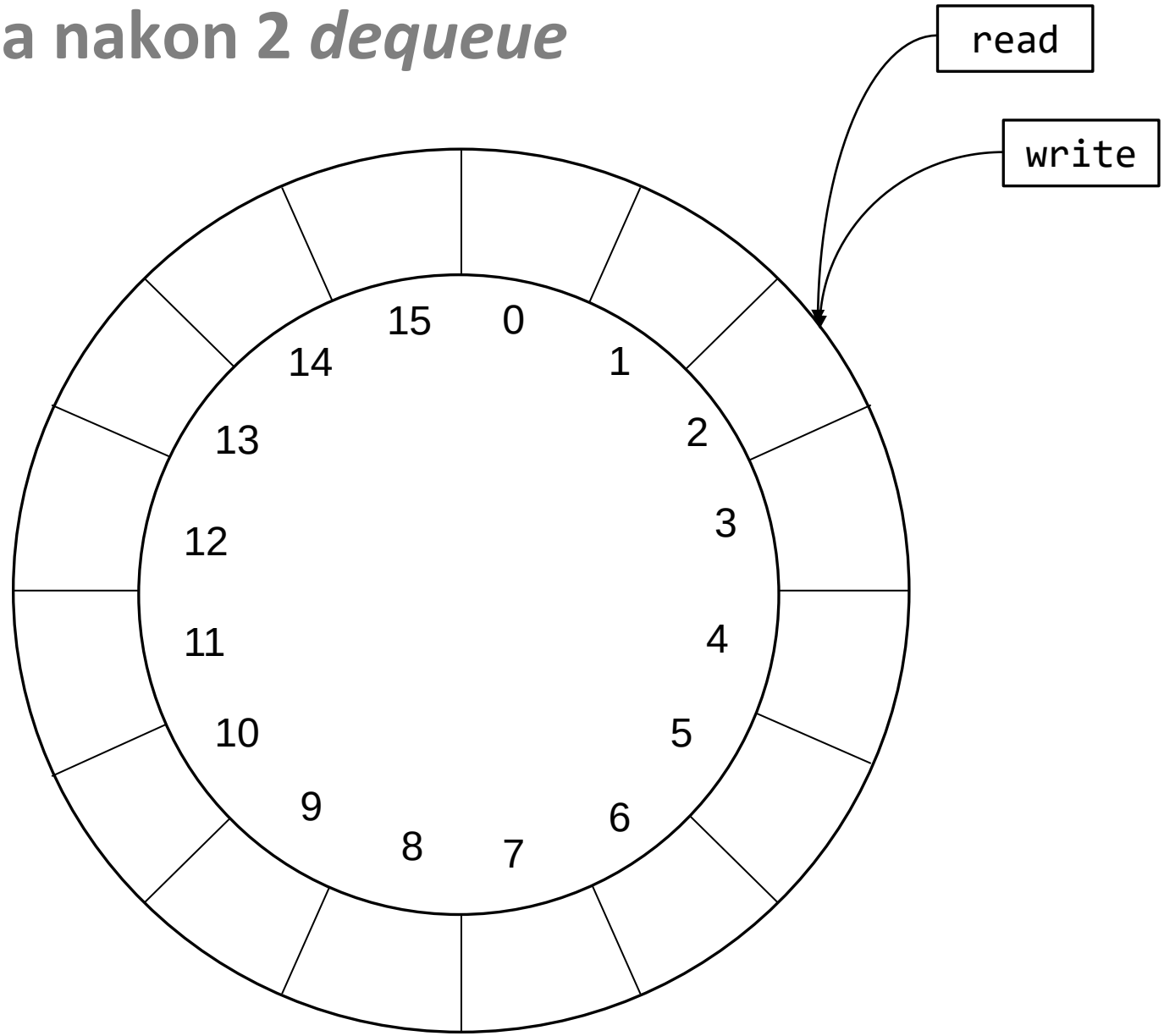
```
i = (i + 1) % n
```



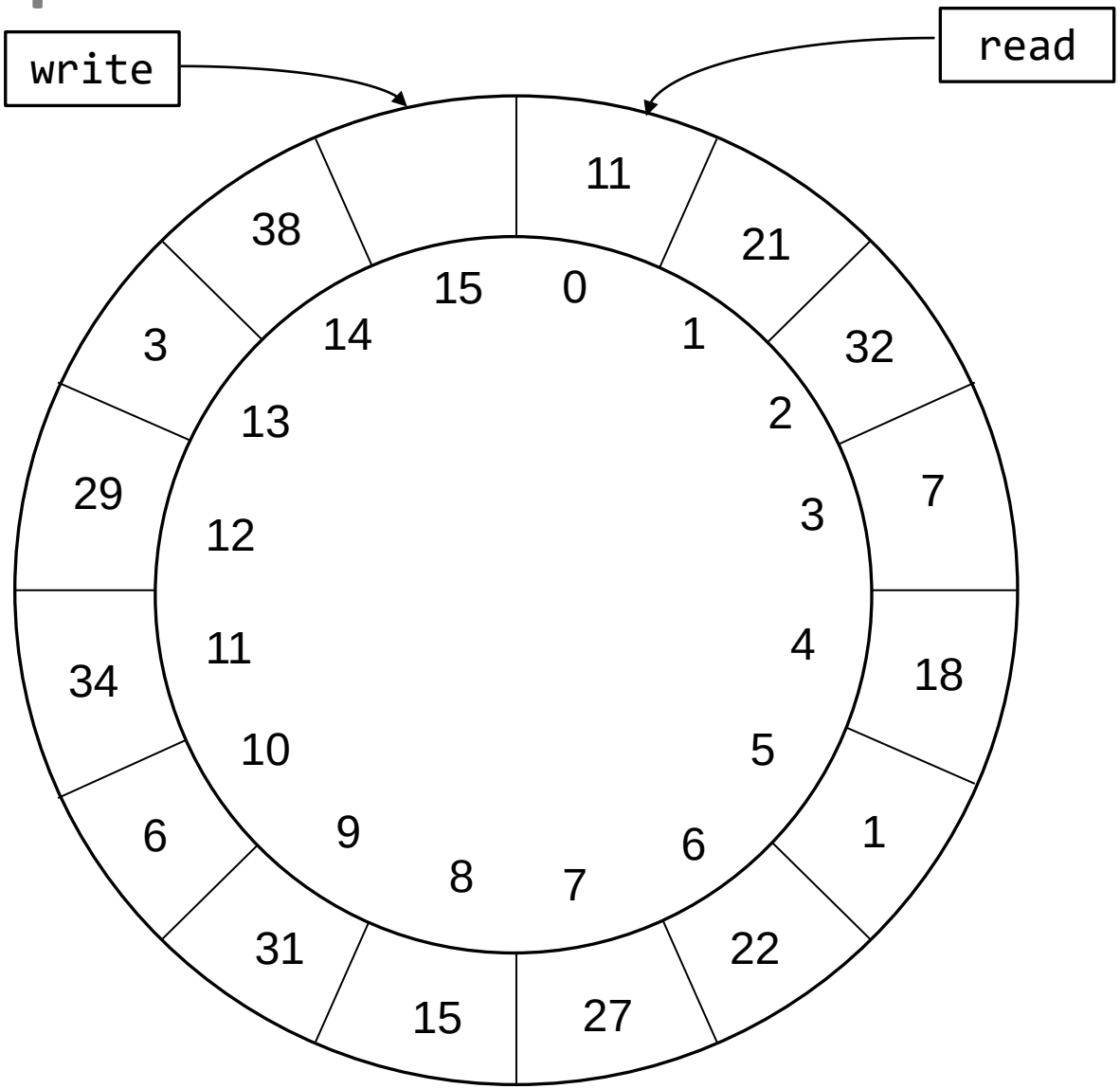
Cirkularna lista s 2 elementa



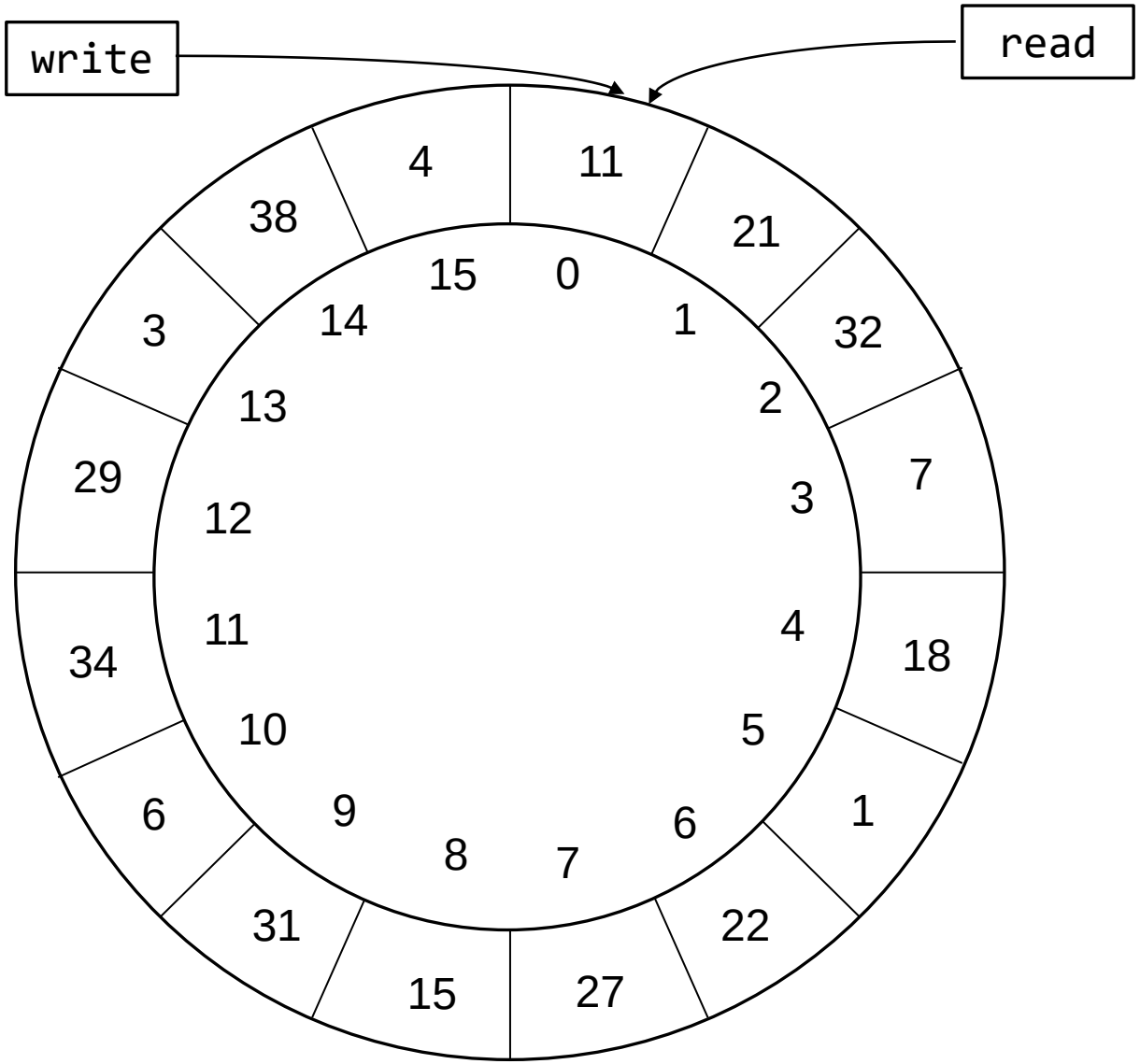
Ista lista nakon 2 *dequeue*



Gotovo puna lista



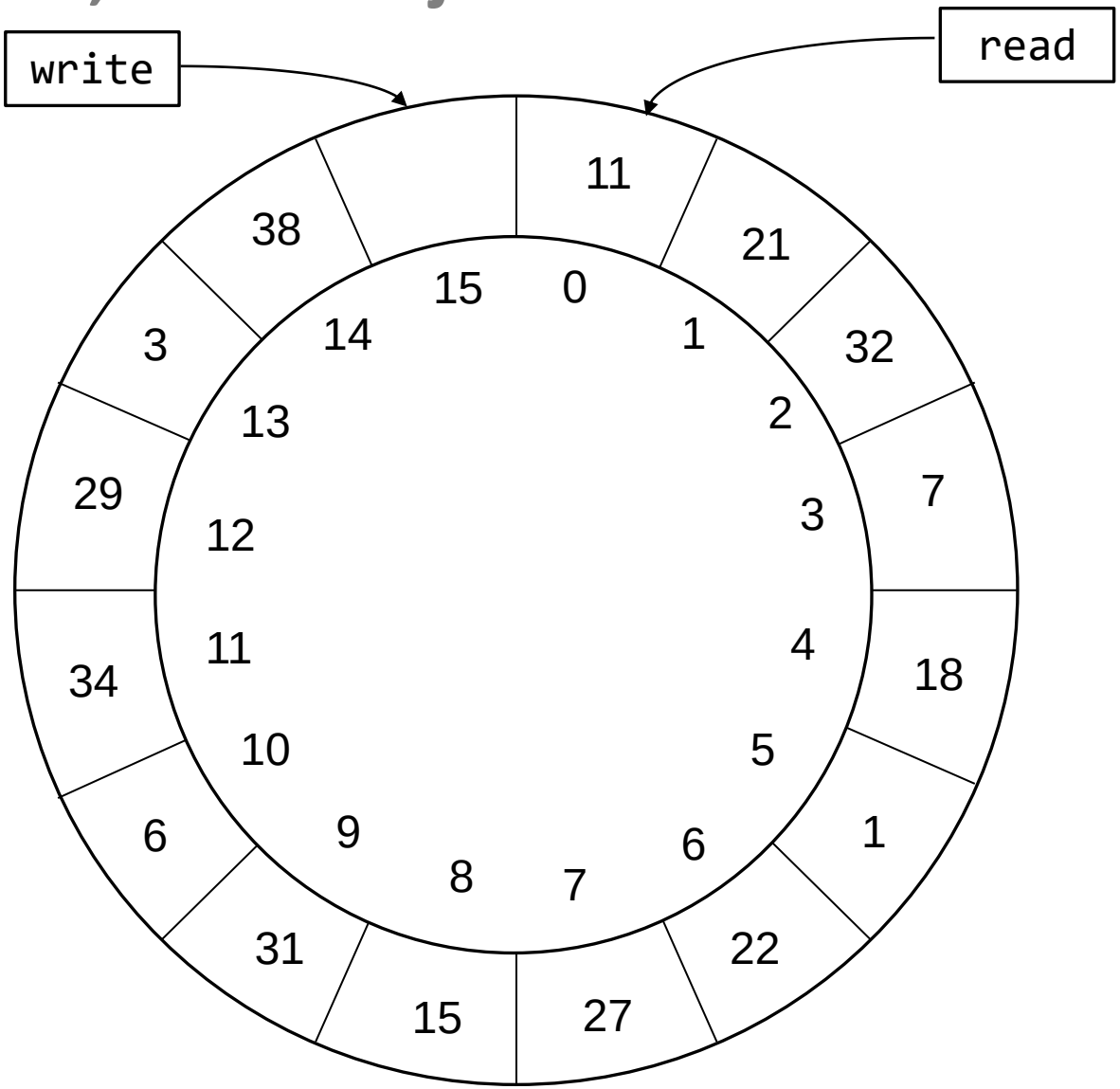
Gotovo puna lista nakon *enqueue*



Cirkularna lista

- Dakle, situacija kad je `read == write` može značiti:
  - praznu listu
  - punu listu
- kako razlikovati?

Puna lista, konvencijom



Kriteriji za praznu i punu listu

- Prazna lista

  `read == write`

- Puna lista

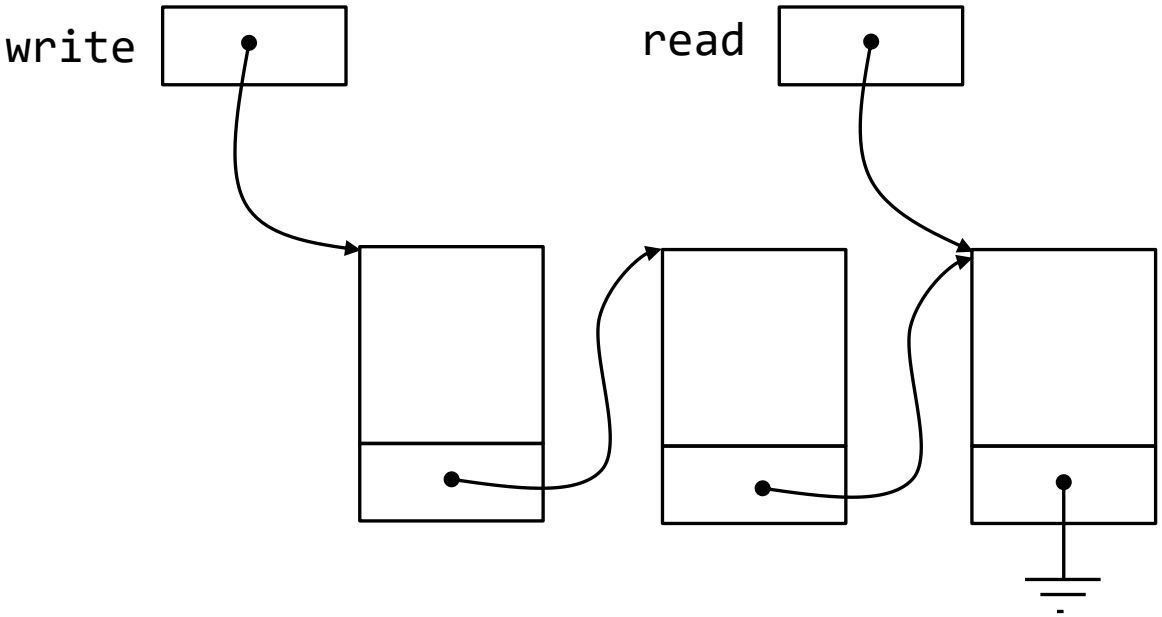
  `(write + 1) % n == read`

- Primjer:

QueueStatic.cpp

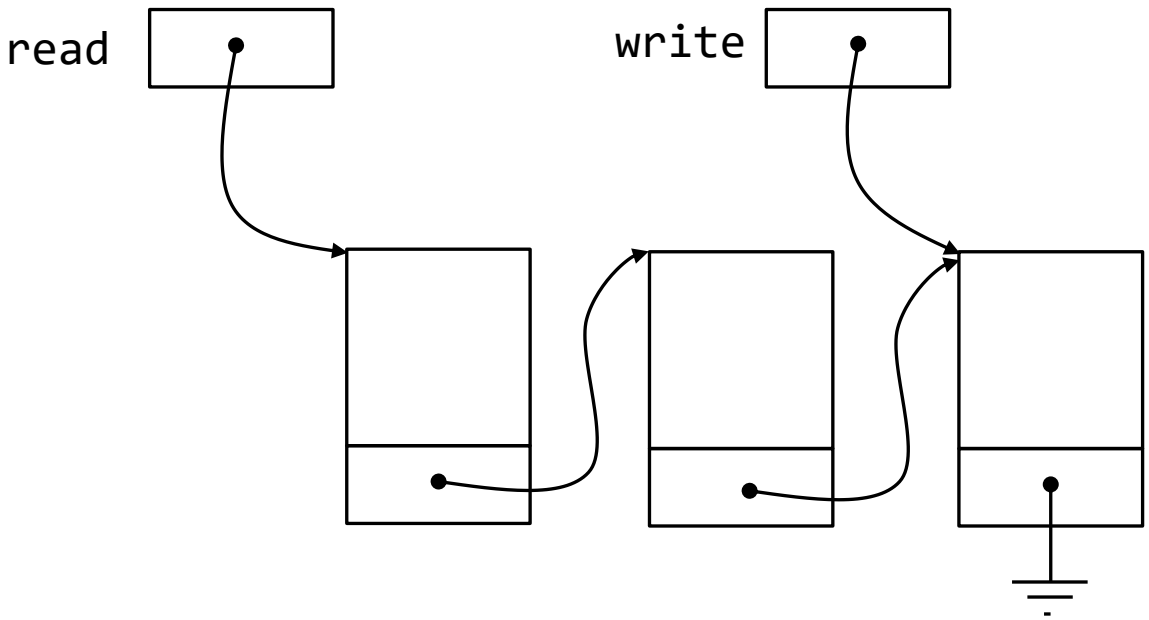
- Najbrža realizacija, O(1), ali postoji mogućnost prepunjenja

Red listom, neispravno



- *enqueue* O(1)
- *dequeue* O(n) (treba `read` postaviti na predzadnjeg)

Red listom, ispravno



- *enqueue* O(1)
- *dequeue* O(1) ☺

QueueList.cpp

Primjer: pretvorba infix izraza u postfix

- Algoritam Shunting-yard
  - analiziraj izraz s lijeva na desno
  - ako slijedi vrijednost – pohrani vrijednost u red
  - ako slijedi operator *o*:
    - dokle god na vrhu stoga postoji operator *o'* višeg ili jednakog prioriteta od *o*
      - skini operator *o'* sa stoga i stavi u red
    - kada to (više) nije slučaj, dodaj operator *o* na stog
  - ako slijedi lijeva zagrada '(', postavi '(' na stog
  - ako slijedi desna zagrada ')', skini sve operatore sa stoga i dodaj u red, sve dok ne dođeš do lijeve zagrade '(', koja se odbacuje
  - ako nema više elemenata u izrazu, sve operatore sa stoga dodaj u red

RPNWithShuntingYard.cpp