The Bakery Algorithm

```
while(1)
{
  choosing[i] = true;
  num[i] = max(num[0], ... , num[n-1]) + 1;
  choosing[i] = false;

  for (j=0; j<n; j++)
  {

     while (choosing[j])
        ;

     while ((num[j] != 0) &&
            ((num[j],j) < (num[i],i)))
        ;

  }

  /* critical section */

  num[i] = 0;
}
```


Reader-Writer Problem 1: Shared variables

semephore read_mutex This allows exclusive access to `read_count`.

semephore writer This allows excusive access to one writer or several readers.

int read_counter This is the number of readers currently sharing the `writer` semephore.

Reader-Writer Problem 1: Writer

```
wait(writer);

/* writer critical section */

signal(writer);
```


Reader-Writer Problem 1: Reader

```
wait(read_mutex)
   if (read_count==0)
      wait(writer);
   read_count++;
signal(read_mutex);

/* reader critical section */

wait(read_mutex);
   read_count--;
   if (read_count==0)
      signal(writer);
signal(read_mutex);
```

Reader-Writer Problem 2: Shared variables

semephore read_mutex, write_mutex This allows exclusive access to `read_count`/`write_count`.

semephore writer This allows excusive access to one writer or several readers.

semephore reader This allows the writing code to stop the readers from getting into the shared `writer` semephone.

int read_counter This is the number of readers currently sharing the `writer` semephore.

int write_counter This is the number of writers currently waiting to get the `writer` semephore.

Reader-Writer Problem 2: Writer

```
wait(write_mutex);
   if (write_count==0)
      wait(readers);
   write_count++;
signal(write_mutex);

wait(writers);

/* critical section */

signal(writers);

wait(write_mutex);
   write_count--;
   if (write_count==0)
      signal(readers);
signal(write_mutex);
```


Reader-Writer Problem 2: Reader

```
wait(reader);
   wait(read_mutex)
      if (read_count==0)
         wait(writer);
      read_count++;
   signal(read_mutex);
signal(reader);

/* reader critical section */

wait(read_mutex);
   read_count--;
   if (read_count==0)
      signal(writer);
signal(read_mutex);
```