```
#ifndef QUEUE_H
#define QUEUE_H

struct queue;
typedef struct queue *queue;

extern queue make_queue(void);
extern void enq(queue q, void *element);
extern void *deq(queue q);
extern int empty(queue q);

#endif
```



```
#include "queue.h"
#include <assert.h>
#include <stdlib.h>

struct queue
{
  /* The number of elements in the queue */
  int size;
  /* The phisical position in contents of the first element.
     If the queue is empty, this can be any position in contents */
  int start;
  /* the physical size of contents in number of elements */
  int capacity;
  void **contents;
};

/* it is silly to make this smaller than around 5 */
#define INITIAL_SIZE 5

queue
make_queue(void)
{
  queue temp = malloc(sizeof(struct queue));
  temp -> size = 0;
  temp -> start = 0;
  temp -> capacity = INITIAL_SIZE;
  temp -> contents = malloc(INITIAL_SIZE * sizeof(void *));
  return temp;
}

int
empty(queue q)
{
  return q -> size == 0;
}

void *
deq(queue q)
{
  /* Don't bother making the capacity smaller */
  void *temp = q -> contents[q -> start];
  assert(!empty(q));
  q -> start = (q -> start + 1) % q -> capacity;
  q -> size--;
  return temp;
}

void
enq(queue q, void *element)
{
  if(q -> size == q -> capacity)
    {
      /* We double the size of contents. */
      void **temp = malloc(2 * q -> capacity * sizeof(void *));
      /* Don't bother preserving the start position. Make it zero */
      int i;
      for(i = 0; !empty(q); i++)
        {
          temp[i] = deq(q);
        }
      free(q -> contents);
      q -> contents = temp;
      q -> start = 0;
      q -> size = q -> capacity;
      q -> capacity *= 2;
    }
  /* Now we know there is room in contents */
  q -> contents[(q -> start + q -> size++) % q -> capacity] = element;
}
```

```
Interface:      2p
   Types des paramètres
   Vraiment impératif
   Abstraction pointeur (sémantique par référence uniforme)

Implémentation: 2p
   Utilisation d'un tampon circulaire
   Respect de l'interface
   Réallocation du tampon

Génie Logiciel: 1p
   Utilisation de "assert"
   Indentation
   Code Bref
   Nom des variables
   etc
```