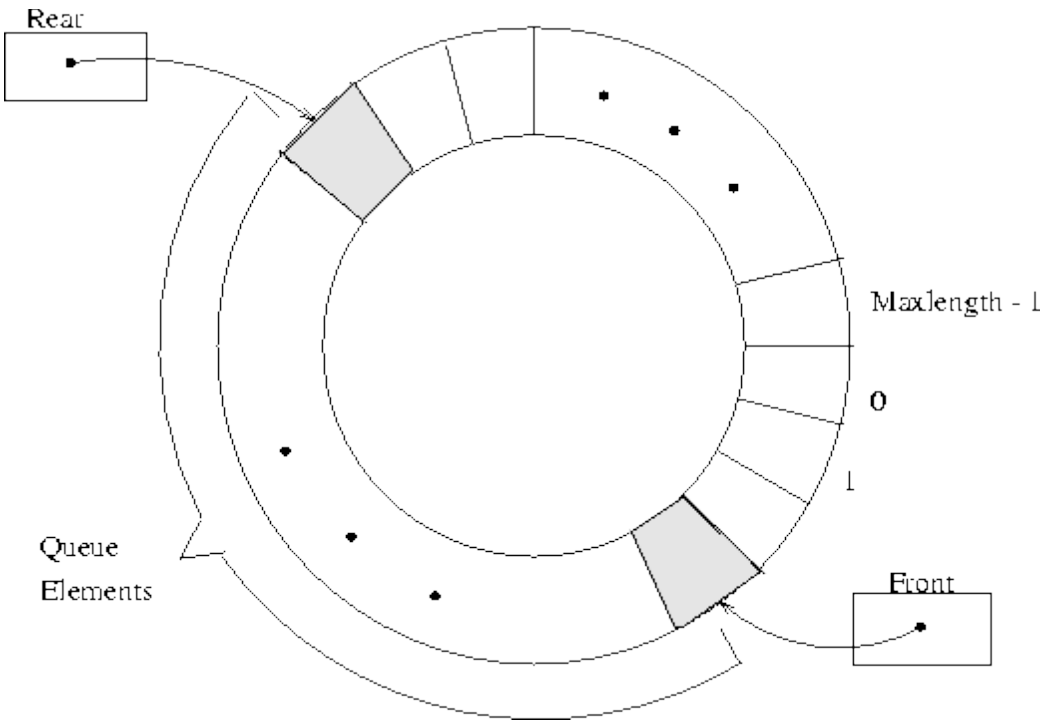
Lab 4 – Pre-processor, header files

Exercise: *typedef, functions, multiple files, conditional compilation*

A queue is a data structure that implements the 'first come, first serve' principle. A queue is also called a FIFO (First in, first out). A queue is an important data structure in many Operating Systems tasks, e.g. in multi-tasking management, print queues, buffering queues (e.g. socket buffers, disk buffers,…), priority queues (see next lab), etc. Write a C program that implements and tests a queue. Use at least 3 files: main.c, queue.c, and queue.h. The queue.c file contains the queue variable to access the queue. Use a **circular array** as underlying data structure to store the queue elements. The array should be defined with **dynamic memory.** At least the following operators should be supported:

- QueueCreate: creates and initializes the queue and prepares it for usage
- QueueDestroy: deletes the queue from memor; the queue can no longer be used unless QueueCreate is called again
- QueueSize: return the number of elements in the queue
- QuequeTop: return the top element in the queue
- Enqueue: add an element to the queue
- Dequeue: remove the top element from the queue



Rear is the position in the queue where new elements are added (enqueue). Front is the position in the queue where elements are removed (top, dequeue).

Use **storage class** specifiers (extern, static, etc.) where needed. For example, to protect access to the queue variable and local functions in queue.c from other files like main.c.

Use **conditional compilation** to allow a user to run the code in 'debug mode'.

Use **Valgrind** to check for potential memory leaks.

Exercise: *GDB*
Compile your code with debug information (How? Check man-pages of gcc). Run gdb to inspect variables, set breakpoints and use the step-by-step execution mode.

Exercise: *pre-processor*
Run the pre-processor (How? Use the man-pages of gcc!) on the code of the previous exercise and save the result in a text file. Open this text file and find out what the pre-processor has done with (i) #include, (ii) #define, and (iii) #ifdef statements.

Exercise: *preprocessor, conditional compilation, gcc toolchain*

THE SOLUTION OF THIS EXERCISE NEEDS TO BE UPLOADED ON TOLEDO BEFORE THE NEXT LAB.

Your solution is only accepted if the following criteria are satisfied:

1. *Compilation with '-Wall -Werror' option generates no warnings or errors*
2. *Running 'cppcheck –-enaheble=all' on the source code results in no warnings or errors*
3. *Using check, a unit-testing framework in c, to validate your solution. Testing with the file myqueue_test.c from Toledo should result in 100%, meaning passing all the tests*

DO NOT upload code that doesn't satisfy these criteria.

Change the queue implementation of the previous exercise such that at compile time the user can set the maximum size of the queue and can choose the data type of the elements of the queue. If the user doesn't define a size and/or a data type at compile time, a default size and data type should be chosen. Only atomic data types can be chosen by the user, i.e. short, int, long, float, double, and char. You may assume that the user will enter its choice in capital letters, e.g. INT or DOUBLE.