1. (ADO client)

An integer stack can be used to convert a positive decimal number *n* to a different numeral system with base *k* according to the following algorithm:

while *n*>0 **do**
  push *n*%*k* onto the stack
  n = *n* / *k*
end while

The result can be displayed by printing the numbers as they are popped off the stack. Example (*k*=2):

```
n = 13          --> push 1 (= 13%2)
n = 6  (= 13/2) --> push 0 (= 6%2)
n = 3  (=  6/2) --> push 1 (= 3%2)
n = 1  (=  3/2) --> push 1 (= 1%2)
n = 0  (=  1/2)
Result: 1101
```

Using your stack ADO from Exercise 5 (Problem Set), write a C-program that implements this algorithm to convert to base *k*=2 a number given on the command line.

Examples of the program executing could be

```
prompt$ ./binary
Enter a number: 13
1101
prompt$ ./binary
Enter a number: 128
10000000
prompt$ ./binary
Enter a number: 127
1111111
```

We have created a script that can automatically test your program. To run this test you can execute the `dryrun` *program that corresponds to this exercise. It expects to find three programs in the current directory:*

- `IntStack.h` *– your header file for the integer stack from Exercise 5 (Problem Set)*
- `IntStack.c` *– your implementation of the integer stack Exercise 5 (Problem Set)*
- `binary.c`

You can use dryrun as follows:

```
prompt$ 9024 dryrun binary
```

2. (Queue ADO)

Modify your integer stack ADO from Exercise 5 (Problem Set) to an integer queue ADO.

Hint: A *queue* is a FIFO data structure (first in, first out). The principal operations are to *enqueue* and to *dequeue* elements. Elements are dequeued in the same order in which they have been enqueued. Below is the header file (`IntQueue.h`) with the functions that your ADO should provide.

`IntQueue.h`

```
// Integer Queue ADO header file ... COMP9024 20T2
#define MAXITEMS 10

void QueueInit();        // set up empty queue
int  QueueIsEmpty();     // check whether queue is empty
void QueueEnqueue(int);  // insert int at end of queue
int  QueueDequeue();     // remove int from front of queue
```

We have created a script that can automatically test your program. To run this test you can execute the `dryrun` *program that corresponds to this exercise. It expects to find two files named* `IntQueue.c` *and* `IntQueue.h` *in the current directory that provide an implementation of a queue ADO with the four queue functions shown above. You can use dryrun as follows:*

```
prompt$ 9024 dryrun IntQueue
```

Submission

This first weekly assignment is meant to give you your first practice and will not count towards your mark for the weekly assessment component.

However, in order to familiarise yourself with the submission and auto-marking process, you can submit your solutions.

You should submit your files using the following **give** command:

```
prompt$ give cs9024 week1 binary.c IntQueue.h IntQueue.c
```

Alternatively, you can submit through WebCMS3.

Ensure that your program compiles on a CSE machine with the standard options `-Wall -Werror -std=c11`.

- Make sure you spell the filenames correctly. You can submit multiple times. Only your last submission will be considered.
- The deadline for submission is **Tuesday, 9 June 11:00:00am**.
- Auto-marking will be run by the lecturer several days after the submission deadline using different test cases than `dryrun` does. *Hint*: Do your own testing in addition to running `dryrun`.