The mechanism for handling petitioners at the Consulate was intended to be a hardware stack; the incoming petitioner would input a 5-bit number (because numbers are in the range 1 to 30), and this number would be pushed further onto the stack as each new petitioner arrived.

When the Consul admitted a petitioner, the top item on the stack would be popped off, and all the others would move up a level.

The stack can be based on the concept of a shift register. Here is a shift register cell which can accept a single bit of information from a higher cell (via input $D_{hi}$, if load is TRUE), or a lower cell (via input $D_{lo}$, if unload is TRUE).

Note that $Q$ is present on two separate outputs for diagrammatic convenience; it allows us to deliver the signal up to a higher element in the stack or down to a lower element without increasing the diagrammatic complexity.

In order to handle 5-bit numbers, this building block can be assembled into a 5-bit-wide storage device.
Note that the *load* and *unload* inputs are common to all elements in the unit.

The next problem is assembling these units into a 30-element stack. That’s easy; they’re organised to slot together. Here’s an abbreviated representation of the previous circuit:

And here we have assembled 30 of the units into a 30-element stack:
Note that the bottom unit has outputs which are not used explicitly, but the Q outputs could be used to check for stack overflow; if a load input occurred when any of the Q outputs from register 29 was 1, then we could signal stack overflow. However, this feature was not specified in the assignment, as it is not required in your circuit.

Here's the complete circuit.
When a new person arrives, and load is made true, their registration number is input into the stack. When the Consul pushes the next button, the current top-of-stack value is captured and loaded into the 5-bit buffer, where it will stay until the next time the next button is pressed, irrespective of any other values that are loaded into the stack in the meantime. The next button also causes the stack to unload, so that the values in the stack move up one position. This does not cause the 5-bit register to get the wrong value, because the stack move and the register load are both synchronous. Therefore the register will load with the value that is in the top-of-stack position when the next signal transits, and the value in the top of stack position will not change till a fraction of a second later.

B

The problem is a simple pedestrian-crisscross problem. It can be solved by designing an ASM which uses a 2-second clock period. The choice of the 2s period allows the shortest state in the traffic light cycle to occur in a single clock cycle, and the other events to be small multiples (2 and 3) of a single clock cycle. It is unnecessary to install separate counters to time the 2, 4, and 6 second periods, as the ASM automatically handles the delays.
Here is a circuit that implements the above ASM. You are not required to use the multiplexor-based model, as in this circuit; a combinatorial version will do.