Solutions to Mid-Term Exam

Question 1

The `for` statement initializes the variable `i` to `N-1` (4), executes the body of the loop while `i` is non-zero and decrements it at the end of each loop. Therefore the values of `i` in the loop will be 4, 3, 2, and 1.

The first expression in the loop sets `c` to the bit-wise logical AND of `i` and 0x02. The value of this expression will be 2 if bit 1 of `i` is set and zero otherwise. The binary values of `i` will be: 100, 011, 010 and 001. Only the second and third values have bit 1 set so the program will print four lines:

```
0
2
2
0
```

Question 2

One possible solution is:

```
/* Count and return the number of blank (space)
   characters in the string s. */

int blanks ( char s[] )
{
   int i, n ;
   n = 0 ;
   for ( i=0 ; s[i] ; i++ ) {
      if ( s[i] == ' ' ) {
         n++ ;
      }
   }
   return n ;
}
```

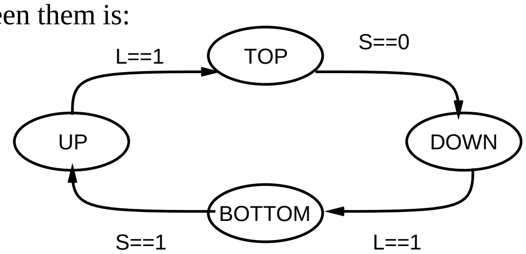
Question 3

As explained in the question, the inputs are the sunshine detector *S*, and the flag limit detector *L*. The two outputs are *U* the motor up/down control and *R*, the motor on/off control.

There are three useful combinations of outputs: off, motor going up, and motor going down. However, if we design the controller using only three states we find that it is not possible to determine when to exit the off state (in one case we need to exit it when the *S* sensor goes high, in the other case we need to exit it when the *S* sensor goes low). We can avoid this problem by using two "off" states: one when the flag reaches the upper limit and one when the flag reaches the lower limit. Appropriate names and outputs might be as shown in the following table:

state	*R*	*U*
TOP	0	0
BOTTOM	0	0
DOWN	1	0
UP	1	1

The state transition diagram showing the states and the logical conditions that cause transitions between them is:

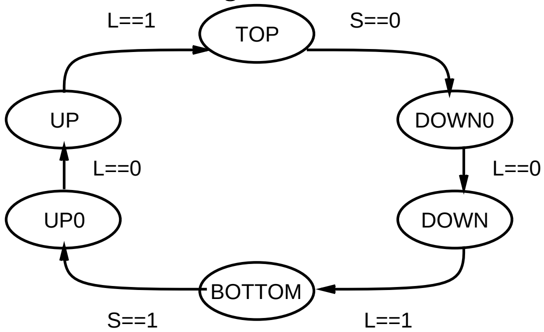


and a tabular description of the state machine is:

starting state	input *S*	input *L*	next state
TOP	0	X	DOWN
TOP	1	X	TOP
DOWN	X	0	DOWN
DOWN	X	1	BOTTOM
BOTTOM	0	X	BOTTOM
BOTTOM	1	X	UP
UP	X	0	UP
UP	X	1	TOP

where the "X" indicates that an *input* has no effect.

However, this solution will not work properly if the state transitions happen so fast that the limit switch will still be active ($L = 1$) after the controller has spent just one clock period in the `UP` or `DOWN` states. In this case the controller would immediately transition from `UP` to `DOWN` to `BOTTOM` states without moving the flag very far. To avoid this problem we can either slow down the state transitions (by using a slow clock) or add two additional states as shown in the state transition diagram below:



for which the state transition table is:

starting state	input S	input L	next state
`TOP`	0	X	`DOWN0`
`TOP`	1	X	`TOP`
`DOWN0`	X	0	`DOWN`
`DOWN0`	X	1	`DOWN0`
`DOWN`	X	0	`DOWN`
`DOWN`	X	1	`BOTTOM`
`BOTTOM`	0	X	`BOTTOM`
`BOTTOM`	1	X	`UP0`
`UP0`	X	0	`UP`
`UP0`	X	1	`UP0`
`UP`	X	0	`UP`
`UP`	X	1	`TOP`

The outputs for the `DOWN0` and `UP0` would be the same as the outputs for the `DOWN` and `UP` states respectively.

Either solution is acceptable.

Question 4

The logic equations for each of the four outputs can be written in sum-of-products form as:

$Y0 = \overline{\mathrm{A}}\,\overline{\mathrm{B}}$

$Y1 = \overline{\mathrm{A}}B$

$Y2 = A\overline{\mathrm{B}}$

$Y3 = AB$

and a schematic diagram for the decoder is: