Add Bees to the Maze

Up to this point you have worked with an illustration of the maze. To describe a room you have had to manually enter the room number.

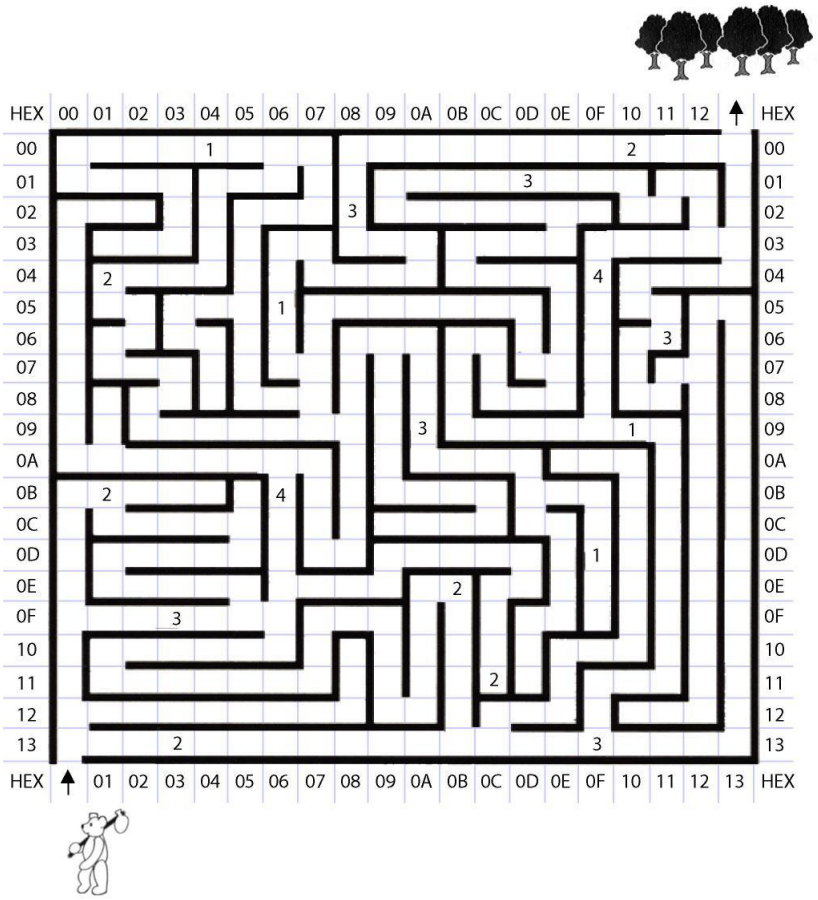

Figure 1: An Illustration of the Maze

In this lab you will begin working with an encoded version of the maze located in a look-up table.

sketch_sep12b | maze.h

```
const int maze_length=280;
// 00   01   02   03   04   05   06   07   08   09   0A   0B   0C   0D   0E   0F   10   11   12   13
const prog_uint8_t theMaze[] PROGMEM =
{0x05,0x09,0x09,0x09,0x09,0x09,0x01,0x03,0x05,0x09,0x09,0x09,0x09,0x09,0x09,0x09,0x29,0x09,0x09,0x02,  // 00
 0x0C,0x09,0x09,0x03,0x05,0x09,0x0A,0x06,0x06,0x05,0x09,0x09,0x09,0x09,0x09,0x09,0x03,0x05,0x03,0x06,  // 01
 0x05,0x09,0x0B,0x06,0x06,0x05,0x09,0x0A,0x06,0x0C,0x09,0x09,0x09,0x09,0x01,0x0B,0x0C,0x0A,0x06,0x06,  // 02
 0x06,0x0D,0x09,0x0A,0x06,0x06,0x05,0x03,0x0C,0x09,0x03,0x05,0x09,0x09,0x0A,0x05,0x09,0x09,0x08,0x02,  // 03
 0x06,0x05,0x09,0x09,0x0A,0x06,0x06,0x0C,0x09,0x09,0x0A,0x0C,0x09,0x09,0x03,0x06,0x05,0x09,0x09,0x0A,  // 04
 0x06,0x0C,0x03,0x05,0x09,0x02,0x06,0x05,0x09,0x09,0x09,0x09,0x09,0x03,0x06,0x06,0x0C,0x03,0x05,0x03,  // 05
 0x06,0x05,0x0A,0x0C,0x03,0x06,0x06,0x06,0x05,0x01,0x03,0x05,0x03,0x06,0x06,0x06,0x05,0x0A,0x06,0x06,  // 06
 0x06,0x0C,0x09,0x03,0x06,0x06,0x0C,0x02,0x06,0x06,0x06,0x06,0x06,0x0C,0x02,0x06,0x06,0x05,0x02,0x06,  // 07
 0x06,0x07,0x05,0x0A,0x0E,0x0C,0x09,0x02,0x06,0x06,0x06,0x06,0x0C,0x09,0x0A,0x06,0x0C,0x0A,0x06,0x06,  // 08
 0x06,0x06,0x0C,0x09,0x09,0x09,0x09,0x08,0x02,0x06,0x36,0x0C,0x09,0x09,0x09,0x08,0x09,0x03,0x06,0x06,  // 09
 0x0C,0x08,0x09,0x09,0x09,0x09,0x01,0x03,0x06,0x06,0x0C,0x09,0x09,0x03,0x0D,0x09,0x03,0x06,0x06,0x06,  // 0A
 0x05,0x01,0x09,0x09,0x0B,0x07,0x06,0x06,0x06,0x0C,0x09,0x09,0x03,0x04,0x09,0x03,0x06,0x06,0x06,0x06,  // 0B
 0x06,0x0C,0x09,0x09,0x09,0x02,0x06,0x06,0x06,0x0D,0x09,0x09,0x0A,0x0C,0x03,0x06,0x06,0x06,0x06,0x06,  // 0C
 0x06,0x05,0x09,0x09,0x09,0x0A,0x06,0x0C,0x0A,0x05,0x09,0x09,0x09,0x03,0x06,0x16,0x06,0x06,0x06,0x06,  // 0D
 0x06,0x0C,0x09,0x09,0x09,0x03,0x04,0x09,0x09,0x0A,0x05,0x23,0x05,0x0A,0x06,0x06,0x06,0x06,0x06,0x06,  // 0E
 0x04,0x09,0x09,0x09,0x09,0x08,0x02,0x05,0x09,0x03,0x06,0x06,0x06,0x05,0x0A,0x0E,0x06,0x06,0x06,0x06,  // 0F
 0x06,0x05,0x09,0x09,0x09,0x09,0x0A,0x06,0x07,0x06,0x06,0x06,0x06,0x06,0x05,0x09,0x0A,0x06,0x06,0x06,  // 10
 0x06,0x0C,0x09,0x09,0x09,0x09,0x09,0x0A,0x06,0x06,0x06,0x06,0x0E,0x0E,0x06,0x05,0x09,0x0A,0x06,0x06,  // 11
 0x04,0x09,0x09,0x09,0x09,0x09,0x09,0x09,0x0A,0x0C,0x0A,0x06,0x05,0x09,0x0A,0x06,0x0D,0x09,0x0A,0x06,  // 12
 0x04,0x09,0x09,0x09,0x09,0x09,0x09,0x09,0x09,0x09,0x09,0x08,0x08,0x09,0x09,0x08,0x09,0x09,0x09,0x0A,  // 13
};
```

Figure 2: An Encoded Maze

The green numbers across the top and right side of the encoded maze (Figure 2) correspond to the columns and rows of the maze. Each entry in the table defines the room at that row and column address. For example; after taking his first step, the robot is in the room at column address 00_{16} and row address 13_{16}. Looking at our maze (Figure 2), we see at these coordinates is a room with only a west facing wall. Figure 3 "Wall Definitions" tells us that a room with only a west facing wall is encoded as 0100_2 = 0x04. Looking at the first entry in the last line of the table, we see our room encoded as 0x04.

0000	1000
0001	1001
0010	1010
0011	1011
0100	1100
0101	1101
0110	1110
0111	1111

Figure 3: Wall Definitions

Each entry in the maze only requires the least significant nibble (4 bits) of each byte in the table. In this table you will be updating the table to include the bees in the maze. For example, room 0x09 at coordinates row = 13, column = 3 has two (2) bees in it (see Figure 1). To include these bees in our table we would change the entry from 0x09 to 0x29.

Question 1

You can find a text version of the maze with rooms only in the Lab04 folder named `maze.inc`. Open this maze in notepad or the Arduino IDE and add the number of bees as defined in Figure 1.

Lookup Tables

In order to keep track of the robot as it physically moves through the maze, we will need to update its orientation and position in the maze as it reaches each intersection. Because the robot does not have any sensors to detect the walls of the maze, you will be creating several functions to update this information.

Question 2

After reading Lab 4 Section "Creating 2-Dimensional turn_table Array" complete the table below and add these tables to the maze.h file.

```
//      Compass     S     E     W     N
//          dir    00    01    10    11
const prog_uint8_t turn_table[] PROGMEM =
{                 0b__,0b__,0b__,0b__   // 00 no turn
                  0b__,0b__,0b__,0b__   // 01 turn right
                  0b__,0b__,0b__,0b__   // 10 turn left
                  0b__,0b__,0b__,0b__   // 11 turn around
```

Question 3

After reading Lab 4 Section "Calculating 1-Dimensional Byte Index" complete the Table 4.0 "How turn_table appears in FLASH program memory" below.

Row		Column		FLASH		Data	Description
Turn		dir		"Byte" Index			
10	2	10	2	10	2		← Base
0	00	0	00	0	0b00000000		no turn and facing South ⇨ ?
0	00	1	01	1	0b00000001		no turn and facing East ⇨ ?
0	00	2	10	2	0b00000010		no turn and facing West ⇨ ?
0	00	3	11	3	0b00000011		no turn and facing North ⇨ ?
1	01	0	00	4	0b00000100		turn right while facing South ⇨ ?
1	01	1	01	5	0b00000101		turn right while facing East ⇨ ?
1	01	2	10	6	0b00000110		turn right while facing West ⇨ ?
1	01	3	11	7	0b00000111		turn right while facing North ⇨ ?
2	10	0	00	8	0b00001000	0b**01**	turn left while facing South ⇨ East
2	10	1	01	9	0b00001001	0b**11**	turn left while facing East ⇨ North
2	10	2	10	10	0b00001010	0b**00**	turn left while facing West ⇨ South
2	10	3	11	11	0b00001011	0b**10**	turn left while facing North ⇨ West
3	11	0	00	12	0b00001100		turn around while facing South ⇨ ?
3	11	1	01	13	0b00001101		turn around while facing East ⇨ ?
3	11	2	10	14	0b00001110		turn around while facing West ⇨ ?
3	11	3	11	15	0b00001111		turn around while facing North ⇨ ?

Table 4: How turn_table appears in FLASH program memory

Question 4

After reading Section “Creating 2-Dimensional map_table Array” complete the table below.

```
//                row col   dir
const prog_uint8_t map_table[] PROGMEM =
{                   __, __   // 00
                   __, __   // 01
                   __, __   // 10
                   __, __   // 11
```

Question 5

After reading Section “Step by Step” complete the Table 6.0 “How map_table appears in FLASH program memory” below.

Robot Walking		FLASH		Data	Description
dir		“Byte” Index			
10	2	10	2		← Base
0	00	0	0b00000000		walking South, ?
0	00	1	0b00000001		walking South, ?
1	01	2	0b00000010	0	walking East, do not increment row
1	01	3	0b00000011	1	walking East, increment column
2	10	4	0b00000100		walking West, ?
2	10	5	0b00000101		walking West, ?
3	11	6	0b00000110		walking North, ?
3	11	7	0b00000111		walking North, ?

Table 6: How map_table appears in FLASH program memory

What Should I Turn In?

Turn in the following material.

1. Title page with the pre-lab number, your name and picture, today’s date, and the day your lab meets.
2. Tables associated with Questions 1 to 5