a)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| H | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 |
| G | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 10 |
| F | | | G/30 | 29 | 28 | 27 | 26 | 9 |
| E | | | | | | | | 8 |
| D | | | 3 | 2 | 4 | 5 | 6 | 7 |
| C | | | | S/1 | | | | |
| B | | | | | | | | |
| A | | | | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

We number the columns and the rows as indicated. The board can be represented as a graph whose nodes are cells of the board, named by their position. For instance, A1, B1,C2, etc. The neighbors of a node are the non-blocked nodes that can be reached using 'up', 'left', 'right', and 'down'. For example, C4 has three neighbors: C5, D4,B4.

The start node is C4 and the goal node is F3.

Starting from C4, we apply the depth-first algorithm with cycle checking.

The frontier: F = {<C4>}.

**Step 1**: C4 is selected. C4 has three neighbors (order according to the rules specified by the exercise): D4,C5,B4
Adding to the frontier gives: F ={<C4,D4>,<C4,C5>,<C4,D4>}

**Step 2**: D4 is selected. It has three neighbors: D3,D5,C4
Adding to the frontier gives: F ={<C4,D4,D3>,<C4,D4,D5>,<C4,C5>,<C4,D4>}
Note that <C4,D4,C4> does not get added because it is eliminated by cycle checking (C4 already occurs previously in the path).

**Step 3**: D3 is selected. It has one neighbor: D4
The path <C4,D4,D3,D4> does not get added to the frontier because of cycle checking. So
The frontier is: F ={<C4,D4,D5>,<C4,C5>,<C4,D4>}

**Step 4**: D5 is selected. It has 3 neighbors: D4,D6,C5
The path <C4,D4,D5,D4> does not get added to the frontier (cycle checking).
The frontier is: F ={<C4,D4,D5,D6>,<C4,D4,D5,C5>,<C4,C5>,<C4,D4>}

**Step 5**: D6 is selected. It has 3 neighbors: D5,D7,C6
The frontier is: F ={<C4,D4,D5,D6,D7>,<C4,D4,D5,D6,C6>,

<C4,D4,D5,C5>,<C4,C5>,<C4,D4>}

**Step 6**: D7 is selected. It has 3 neighbors: D6,D7,D8
The frontier is: F ={<C4,D4,D5,D6,D7,D8>,<C4,D4,D5,D6,C6>,
<C4,D4,D5,C5>,<C4,C5>,<C4,D4>}

**Step 7**: D8 is selected. It has 2 neighbors: E8,C8
The frontier is: F ={<C4,D4,D5,D6,D7,D8,E8>,<C4,D4,D5,D6,D7,D8,C8>
<C4,D4,D5,D5,C6>,<C4,D4,D5,C5>,<C4,C5>,<C4,D4>}

**Step 8** to **Step 11**: We move from D8 to H8
**Step 12** to **Step 18**: We move from H7 to H1
**Step 19** to **Step 25**: We move from G1 to G7
**Step 26** to **Step 30**: We move from G6 to G3

The sequence of nodes is given by the red numbers written inside them.

b)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| H | 4 | 3 | 2 | 3 | 4 | 5 | 6 | 7 |
| G | 3 | 2 | 1 | 2 | 3 | 4 | 5 | 6 |
| F | 2/13 | 1/14 | G/15 | 1 | 2 | 3 | 4 | 5 |
| E | 3/12 | 2 | | | | | | 6 |
| D | 4/11 | | 2/3 | 3/2 | 4/4 | 5/5 | 6 | 7 |
| C | 5/10 | 4/9 | | S/4/1 | 5 | 6 | | 8 |
| B | 6 | 5/8 | 4/7 | 5/6 | 6 | 7 | 8 | 9 |
| A | 7 | 6 | 5 | 6 | 7 | 8 | 9 | 10 |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

The Manhattan distance table and best-first with multiple-path pruning
(Distance is in red, Movement in blue)

The frontier: F = {<C4>}.
**Step 1**: C4 is selected. C4 has three neighbors (order by the distance): D4,C5,B4
Adding to the frontier gives: F ={<C4,D4>,<C4,C5>,<C4,B4>}
**Step 2**: D4 is selected. It has three neighbors: D3,D5,C4
Adding to the frontier gives: F ={<C4,D4,D3>,<C4,D4,D5>,<C4,C5>,<C4,B4>}
<C4,D4,C4> does not get added because of multi-path pruning.
**Step 3**: D3 is selected. It has one neighbor: D4
The path <C4,D4,D3,D4> does not get added to the frontier because of multi-path

pruning which subsumes cycle checking. So
The frontier is: F ={<C4,D4,D5>,<C4,C5>,<C4,B4>}

**Step 4**: D5 is selected. It has 3 neighbors: D4,D6,C5
The path <C4,D4,D5,D4> does not get added to the frontier (cycle checking).
The frontier is: F ={<C4,D4,D5,D6>,<C4,D4,D5,C5>,<C4,C5>,<C4,B4>}

**Step 5**: Here, we can select either D6, C5, or B4.
D6 is selected. It has 3 neighbors: D5,D7,C6
The frontier is: F ={<C4,D4,D5,D6,D7>,<C4,D4,D5,D6,C6>,
<C4,D4,D5,C5>,<C4,C5>,<C4,B4>}

**Step 6**: Here, we can select either C5 or B4. (Selecting C5 will later make us select B4.)
B4 is selected. It has 3 neighbors: D5,D7,C6
The frontier is: F ={<C4,D4,D5,D6,D7>,<C4,D4,D5,D6,C6>,
<C4,D4,D5,C5>,<C4,C5>,<C4,B4,B3>,<C4,B4,A4>,
<C4,B4,B5>}

Continue like that, we have the nodes visited as depicted in the picture (in blue).

c)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| H | 4 | 3 | 2 | 3 | 4 | 5 | 6 | 7 |
| G | 3 | 2 | 1 | 2 | 3 | 4 | 5 | 6 |
| F | 2 | 1 | G/14 | 1/13 | 2/12 | 3/11 | 4/10 | 5/9 |
| E | 3 | 2 | | | | | | 6/8 |
| D | 4 | | 2/3 | 3/2 | 4/4 | 5/5 | 6/6 | 7/7 |
| C | 5 | 4 | | S/4/1 | 5 | 6 | | 8 |
| B | 6 | 5 | 4 | 5 | 6 | 7 | 8 | 9 |
| A | 7 | 6 | 5 | 6 | 7 | 8 | 9 | 10 |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

The Manhattan distance table and heuristic depth-first with cycle checking
(Distance is in red, Movement in blue)

d)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| H | 4 | 3 | 2 | 3 | 4 | 5 | 6 | 7 |
| G | 3 | 2 | 1 | 2 | 3 | 4 | 5 | 6 |
| F | 8/2 | 9/1/19 | G/20 | 1 | 2 | 3 | 4 | 5 |
| E | 7/3/17 | 8/2/18 | | | | | | 6 |
| D | 6/4/16 | | 2/2/3 | 1/3/2 | 2/4/4 | 3/5/5 | 4/6 | 7 |
| C | 5/5/15 | 4/4/10 | | S/0/4/1 | 1/5/6 | 2/6/11 | | 8 |
| B | 4/6 | 3/5/9 | 2/4/8 | 1/5/7 | 2/6/12 | 3/7 | 8 | 9 |
| A | 7 | 4/6 | 3/5/14 | 2/6/13 | 3/7 | 8 | 9 | 10 |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

The Manhattan distance table and A*
(Distance is in red, Movement in blue)

Some notes:

- I do include cycle checking
- X/Y/Z:
  - X is the actual cost (number of movements from start node to the node)
  - Y is the Manhattan distance
  - Z is the order the node is visited
- In the third step, we can select either D5, C5, or B4

e)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| H | 4 | 3 | 2 | 3 | 4 | 5 | 6 | 7 |
| G | 3 | 2 | 1 | 2 | 3 | 4 | 5 | 6 |
| F | 2 | 1/10 | G/11 | 1 | 2 | 3 | 4 | 5 |
| E | 3/8 | 2/9 | | | | | | 6 |
| D | 4/7 | | 12 | 11 | 10 | 9 | 8 | 7 |
| C | 5/6 | 6/5 | | S/10/1 | 11 | 10 | | 8 |
| B | 6 | 7/4 | 8/3 | 9/2 | 10 | 11 | 10 | 9 |
| A | 7 | 8 | 9 | 10 | 11 | 12 | 11 | 10 |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

*dist* table for dynamic programming (Distance is in red, Movement in blue)

f) I leave it to you J