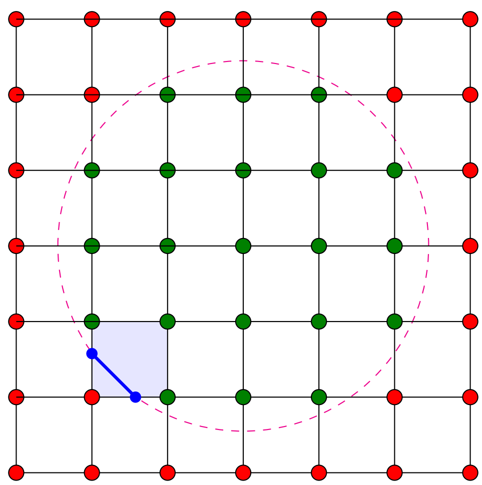
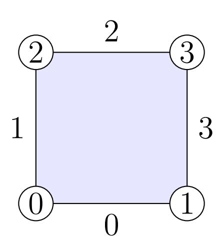
CS130 - LAB - Marching Squares

Name: ______________________________ SID: _________________

In this lab we will generate contours in a 2-dimensional field using a method called marching squares. The input is a matrix with positive and negative numbers and we want to draw a isoline where the value in the field is zero. In marching squares, we slide a 2×2 window across the grid, compute a 4-bit index for each element in the window (1 if the element is positive, 0 otherwise), and use the index to lookup the line to be draw on a precomputed table. In the figure below, green is inside and red is outside. The shaded square corresponds to case 1, and our approximation of the surface in this square is the blue segment. Notice that the endpoints of this segment lie on the edges of the shaded square.

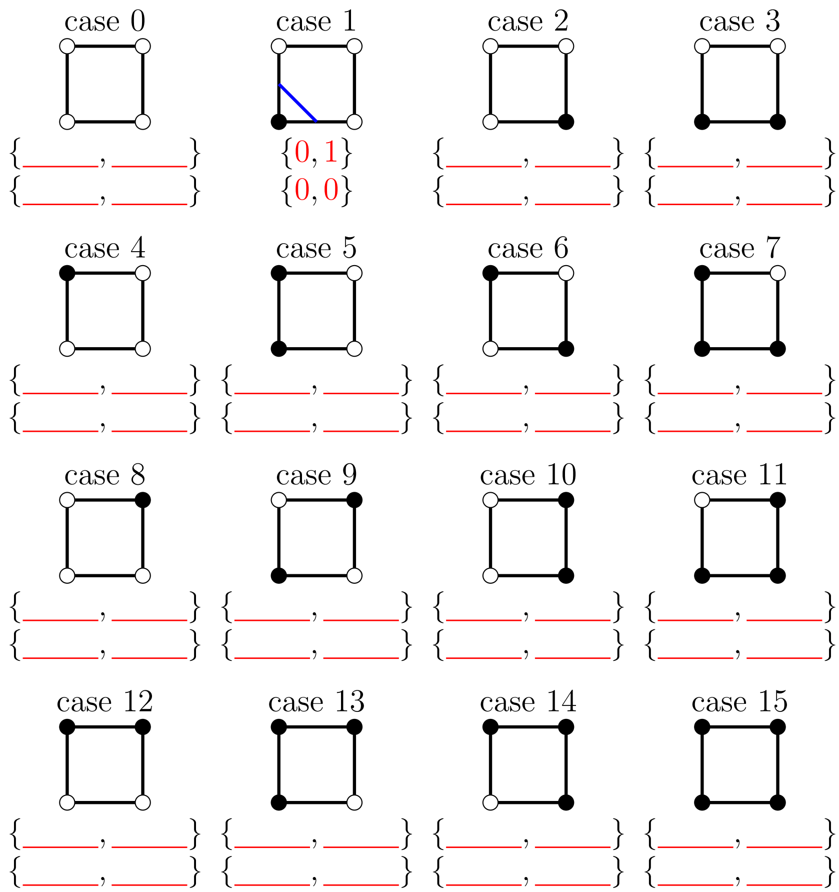


We will use the ordering of nodes and edges shown below.



1. Complete the drawing below with all the segments you need to insert in the table. Fill in the blanks with the endpoints of the segments, which are located on the edges of

the square (the edges of the square are numbered above). If you need fewer than two segments, use $\{0, 0\}$ to indicate that the segment is not required.



2. Fill out the lookup table called `case_table`.

3. Implement the case computation in `compute_case`.

4. Implement the TODO section of `marching_squares`. You can place the endpoints of your segments at the midpoints of the edges of the square at first.

5. Run your code in tests `00.txt` through `04.txt` to test your code. For example, "`./squares 00.txt`" will run the first test case. This will generate the file `output.eps`, which contains an visualization of the surface you generated in blue. The first four test cases should produce diamonds or circles. The last test is a stress test that exercises all of the cases. The blue segments should form closed loops (or curves whose endpoints lie on the sides of the the image). The blue curves should separate the green dots from the red dots.

6. Interpolate along the edges of the square using the scalar values stored at the vertices to get a more accurate placement of the endpoints of the segment. If done correctly,

test `03.txt` should produce a smooth circle. Let ϕ_0 and ϕ_1 be the scalars at the endpoints (you may assume they differ in sign). We want to compute a $0 \le \lambda \le 1$ from ϕ_0 and ϕ_1 so that we can interpolate the locations of the endpoints: $\mathbf{x} = \lambda \mathbf{x}_0 + (1-\lambda)\mathbf{x}_1$. We will assume the form $\lambda = \frac{a\phi_0 + b\phi_1}{c\phi_0 + d\phi_1}$ for some a, b, c, d. The interpolation should have the properties that (1) $\mathbf{x} = \mathbf{x}_0$ when $\phi_0 = 0$ and $\phi_1 \neq 0$, (2) $\mathbf{x} = \mathbf{x}_1$ when $\phi_1 = 0$ and $\phi_0 \neq 0$, and (3) if $\phi_0 = -\phi_1$ then $\mathbf{x} = \frac{\mathbf{x}_0 + \mathbf{x}_1}{2}$. Fill in your interpolation formula here: $\lambda = \frac{\_\_\_\_\_\phi_0 + \_\_\_\_\_\phi_1}{\_\_\_\_\_\phi_0 + \_\_\_\_\_\phi_1}$.