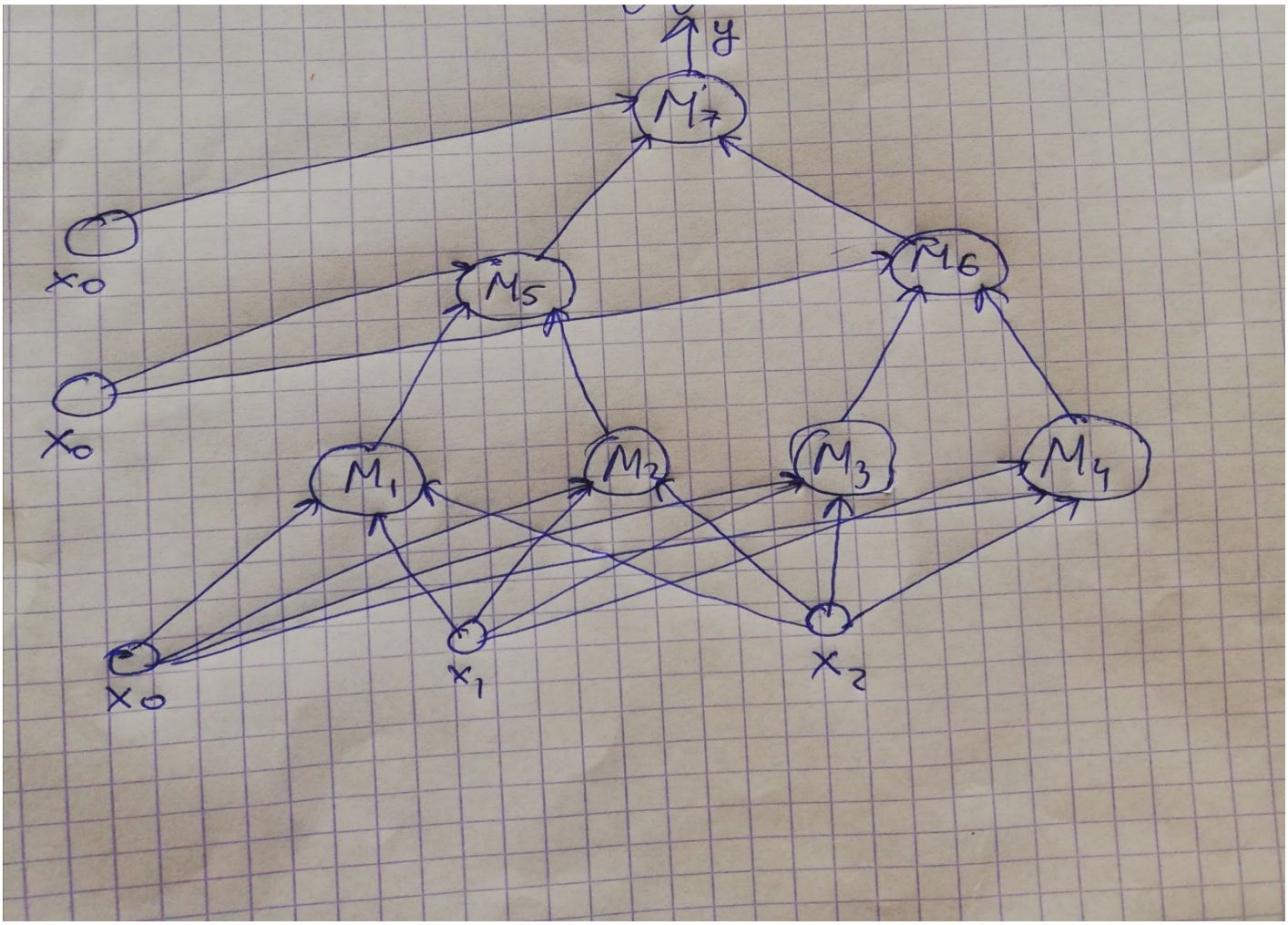
TASK 2

Task 2.3:

For this task I have hard-coded the actual weights for the weight matrices. For the first layer, I found the equation between points 1 and 2,2 and 3,3 and 4 ,and 4 and 1. Then, I find the equations of the lines by finding first the slope of the line and then using the equation y - y0 = m*(x-x0) where m is the slope ,and x0,y0 are one of the coordinates of the points on this line. Then depending on which side of line we want(this is the inner of the polygon) i find w2,w1,w0 such that w2*x2 + w1*x1 + w0 >0. Then I have to normalize them as it is a requirement of the coursework. To get the inner region I need to get an intersection of all of the regions from the four perceptrons used above. I group the first with the second and the third with the forth. I find the intersections between those pairs and then I find the region required by intersecting the regions calculated before. To find that, I need to also find the weight matrix for the AND gate. This is easy :any line that divides the plane in a way that only (1,1) will be above the line, will do the job.This is because we only get 1 if both inputs are 1 in an and gate. I chose the line y = -x +1.5. From there I find the weight matrix for the AND operation. The structure of the neural network is shown below :



M1-4 classify the lines. M5-7 are doing the AND logical gate. Y is the final output

Task 2.10:

In this task I investigate the differences between using hNeurons and sNeurons. At first, by seeing the plot of them from the previous tasks they look identical. I took the same neural network from task 2.6 and implemented it in task 2.8. For that I needed to adjust the weight matrices. For the first layer, which is where we find the weight matrices for the lines, I used the same weights but multiplied them by 10^8. This would cause an overflow and I would get a good result of 0 or 1 depending on the classification. So,after the first layer, the outputs for both are the same. However, in the other layers in which I have to implement logic gates, it was really hard to find weights to make it exactly 0 or 1. Instead, by using sNeurons and the correct weights, I got weights close to 0 or 1 but never 0 or 1 exactly for all of them. This is where the implementation with hNeurons differs as after every layer we would get a 0 or 1. To implement the weight matrices with sNeurons I calculated them by hand so that I would get proximate results to what I was looking for. Nevertheless, the classifications are not the same. They look similar but when I decided to check how many points are classified the same, I was shocked by the results. I used 160000 points in my classification and out of those, 37480 were classified incorrectly. This means that a forth of the results were classified differently from the classification with hNeurons.