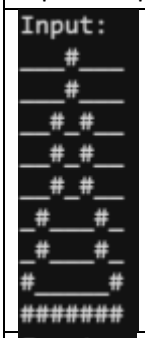
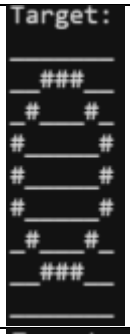
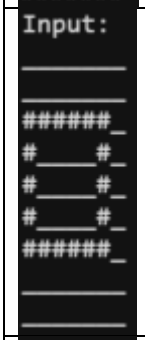
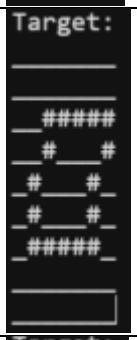
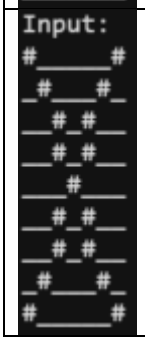
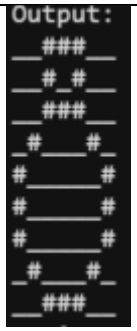
Question 1:

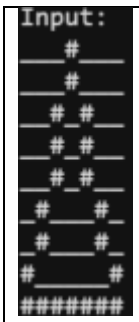
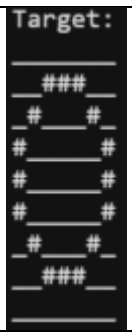
Part a:

Note: the ‘vis’ function defined in my code is applied to the output images to make them easier to visually compare the function turns pixels from integers to symbols based off the following conversion:

1 = #, -1 = _, 0 = *

Input sample image:	Expected target output:	Network output:
		
		
		

Part b:

Input: _______ _______ ######_ #____#_ #____#_ #____#_ ######_ _______ _______	Target: _______ _______ __##### __#___# _#___#_ _#___#_ _#####_ _______ _______	Output: _______ _______ __##### __#___# _#___#_ _#___#_ _#####_ _______ _______
Input: #_____# _#___#_ __#_#__ __#_#__ ___#___ __#_#__ __#_#__ _#___#_ #_____#	Target: __###__ __#_#__ __###__ _#___#_ #_____# #_____# #_____# _#___#_ __###__	Output: __###__ __#_#__ __###__ _#___#_ #_____# #_____# #_____# _#___#_ __###__
Input: _#####_ _#___#_ _#___#_ #_#_#_# _#___#_ #_#_#_# _#___#_ _#___#_ _#####_	Target: _______ _#####_ #_____# #_###_# #_#_#_# #_###_# #_____# _#####_ _______	Output: _______ _#####_ #_____# #_###_# #_#_#_# #_###_# #_____# _#####_ _______
Input: ___#___ #_###_# #_###_# ___#___ ##___## ___#___ #_###_# #_###_# ___#___	Target: _#####_ #_____# _#####_ _______ _______ _______ _#####_ #_____# _#####_	Output: _#####_ #_____# _#####_ _______ _______ _______ _#####_ #_____# _#####_

Part c:

Input: ___**__ __*#*__ __#_*_* __#_#__ __*_#__ _*___#* _#___#_ *___*_# ####*##	Target: _______ __###__ _#___#_ #_____# #_____# #_____# _#___#_ __###__	Output: _______ __###__ _#___#_ #_____# #_____# #_____# _#___#_ __###__ _______
Input: _______ _______ ######_ #____#_ #____#_ #____#_ ######_ _______ _______	Target: _______ _______ __##### __#___# _#___#_ _#___#_ _#####_ _______ _______	Output: _______ _______ __##### __#___# _#___#_ _#___#_ _#####_ _______ _______
Input: #_*___# _#___#_ __#_*__ __#_#__ ___#__* __#*#__ **#*#*_ _***_#_ #__*__#	Target: __###__ __#_#__ __###__ _#___#_ #_____# #_____# #_____# _#___#_ __###__	Output: __###__ __#_#__ __###__ _#___#_ #_____# #_____# #_____# _#___#_ __###__
Input: _#####_ _#___#_ _#___#_ #_#_#_# _#___#_ #_#_#_# _#___#_ _#___#_ _#####_	Target: _______ _#####_ #_____# #_###_# #_#_#_# #_###_# #_____# _#####_ _______	Output: _______ _#####_ #_____# #_###_# #_#_#_# #_###_# #_____# _#####_ _______
Input: ___#___ #_###_# #_###_# ****__* ##__*## *_*#___ *_#*#_# #_#**_# ___#___	Target: _#####_ #_____# _#####_ _______ _______ _______ _#####_ #_____# _#####_	Output: _#####_ #_____# _#####_ _______ _______ _______ _#####_ #_____# _#####_

Question 2:

Part a:

4 separate images can be stored with perfect recall, for example: 1, 0 ,7 ,5. The limit does depend on which images are chosen to be stored, if the images in the set share a large percentage of their pixels, the network will be unable to store as many, for example if the net is trained on 6, 0, 8, 5, it is only able to perfectly recall 1 of them.

Part b:

When the network is trained on 4 images: 1, 0, 7, 5, the network can handle 11 pixels flipped to zeros per image and still have perfect recall of all of them.

The network's noise tolerance does depend on the number of images stored, for example when 3 images are stored such as: 1,0,5, the network is able to perfectly recall these images when 49 pixels are flipped to zeros per image. With just two stored images, 1,0, the network can handle 59 flipped pixels.

Part c:

I performed the re-iteration training on a network trained on 6, 0, 8, 5, the performance of the network improved from perfectly recalling only 1/4 to 2/4.