

**Ain Shams University**

**Faculty Of Engineering**

CSE 365

Computer Vision

OCR Final Project

*Submitted By:*

Andrew Sameh Labib 16P6007

Nayer Nabil 16P6054

Contents

[Solution : 4](#_Toc28099801)

[Steps to reach the output : 8](#_Toc28099802)

[Test Cases 1 : 9](#_Toc28099803)

[Output for testcase 1: 10](#_Toc28099804)

[Test Case 2 : 12](#_Toc28099805)

[Test Case 2 Output : 13](#_Toc28099806)

[Test Case 3 : 15](#_Toc28099807)

[Test Case 3 Output : 16](#_Toc28099808)

List of Figures:

[Figure 1 Test Sample 1 9](#_Toc28099832)

[Figure 2 Test Case 1 Output File 10](#_Toc28099833)

[Figure 3 Test Case 1 Output File (Cont.) 11](#_Toc28099834)

[Figure 4 Test Sample 2 12](#_Toc28099835)

[Figure 5 Test Case 2 Output File 13](#_Toc28099836)

[Figure 6 Test Case 2 Output File (Cont.) 14](#_Toc28099837)

[Figure 7 Test Sample 3 15](#_Toc28099838)

[Figure 8 Test Case 3 Output File 16](#_Toc28099839)

[Figure 9 Test Case 3 Output File (Cont.) 17](#_Toc28099840)

# Solution :

import cv2  
import numpy as np  
import math  
  
  
def rotateImage(image, angle): # image rotation function by calculating the center of the image and then rotate it  
 image\_center = tuple(np.array(image.shape[1::-1]) / 2) # calculating the center of image  
 rot\_mat = cv2.getRotationMatrix2D(image\_center, angle, 1.0) # calculating the suitable rotation matrix  
 result = cv2.warpAffine(image, rot\_mat, image.shape[1::-1], flags=cv2.INTER\_LINEAR) # Getting the final image as a result  
 return result  
  
  
file = open("Output.txt","w+") # Opening the output file to use it  
img = cv2.imread('test\_sample2.jpg', 0) # Reading the input image  
img = cv2.resize(img,(481,680)) # Resizing the image  
img = cv2.bitwise\_not(img) # Inverting the image to be ready for applying some operations  
  
img\_edges = cv2.Canny(img, 100, 100, apertureSize=3) # Edge detection  
lines = cv2.HoughLinesP(img\_edges, 1, math.pi / 180.0, 100, minLineLength=50, maxLineGap=5) # Line detection  
angles = []  
for x1, y1, x2, y2 in lines[0]:  
  
 angle = math.degrees(math.atan2(y2 - y1, x2 - x1)) # Calculating the angle of rotation by calculating slope  
 angles.append(angle)  
  
median\_angle = np.median(angles) # Getting the angle value to rotate the image  
  
img = rotateImage(img,median\_angle) # Image rotation function by passing the image and the angle calculated before  
img = cv2.resize(img,(481,680)) # Resizing again after rotation  
  
ret, thresh = cv2.threshold(img,235,255,cv2.THRESH\_BINARY) # Applying thresholding on the image to get the needed information only  
kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE,(3,3))  
eroded = cv2.erode(thresh,kernel)  
eroded = cv2.dilate(eroded,kernel) # erosion to make sure there is no noise  
  
answers = cv2.connectedComponentsWithStats(eroded, 4, cv2.CV\_32S) # Function to detect the components in the image after thresholding  
answers = answers[2]  
answers = answers[1:23]  
j = 0  
Question\_no = "1.1","1.2","1.3","1.4","1.5","2.1","2.2","2.3","2.4","2.5","2.6","3.1","3.2","3.3","4.1","4.2","4.3",\  
 "5.1","5.2"  
for i in answers: # printing the answers detected from the image  
 if i[1] == answers[0][1]:  
 if (i[0] >= 353) & (i[0] <= 391):  
 file.write("Gender: Male \r\n")  
 elif (i[0] >= 392) & (i[0] <= 446):  
 file.write("Gender: Female \r\n")  
 if i[1] == answers[1][1]:  
 # Continue semester values  
 if (i[0] >= 145) & (i[0] <= 185):  
 file.write("Semester: Fall \r\n")  
 if (i[0] >= 190) & (i[0] <= 260):  
 file.write("Semester: Spring \r\n")  
 if (i[0] >= 290) & (i[0] <= 350):  
 file.write("Semester: Summer \r\n")  
 if i[1] == answers[2][1]:  
 # Program value  
 if (i[0] >= 120) & (i[0] <= 150) & ~(i[1] >= 140):  
 file.write("Program: MCTA \r\n====================================================================== \r\n"  
 "====================================================================== \r\n") # done  
 if (i[0] >= 159) & (i[0] <= 189) & ~(i[1] >= 140):  
 file.write("Program: ENVER \r\n====================================================================== \r\n"  
 "====================================================================== \r\n") # done  
 if (i[0] >= 198) & (i[0] <= 228) & ~(i[1] >= 140):  
 file.write("Program: BLDG \r\n====================================================================== \r\n"  
 "====================================================================== \r\n") # done  
 if (i[0] >= 237) & (i[0] <= 277) & ~(i[1] >= 140):  
 file.write("Program: CESS \r\n====================================================================== \r\n"  
 "====================================================================== \r\n") # done  
 if (i[0] >= 275) & (i[0] <= 305) & ~(i[1] >= 140):  
 file.write("Program: ERGY \r\n====================================================================== \r\n"  
 "====================================================================== \r\n") # done  
 if (i[0] >= 307) & (i[0] <= 337) & ~(i[1] >= 140):  
 file.write("Program: COMM \r\n====================================================================== \r\n"  
 "====================================================================== \r\n") # done  
 if (i[0] >= 354) & (i[0] <= 384) & ~(i[1] >= 140):  
 file.write("Program: MANF \r\n====================================================================== \r\n"  
 "====================================================================== \r\n") # done  
 if (i[0] >= 120) & (i[0] <= 150) & (i[1] >= 140):  
 file.write("Program: LAAR \r\n====================================================================== \r\n"  
 "====================================================================== \r\n") # done  
 if (i[0] >= 159) & (i[0] <= 189) & (i[1] >= 140):  
 file.write("Program: MATL \r\n====================================================================== \r\n"  
 "====================================================================== \r\n") # done  
 if (i[0] >= 198) & (i[0] <= 228) & (i[1] >= 140):  
 file.write("Program: CISE \r\n====================================================================== \r\n"  
 "====================================================================== \r\n") # done  
 if (i[0] >= 237) & (i[0] <= 277) & (i[1] >= 140):  
 file.write("Program: HAUD \r\n====================================================================== \r\n"  
 "====================================================================== \r\n") # done  
  
 while (j != 19) & (i[1] >= answers[3][1]):  
 if i[1] == answers[j+3][1]:  
 # Questions  
 if (i[0] >= 310) & (i[0] <= 340):  
 file.write("Question"+Question\_no[j] + " : Strongly Agree \r\n====================================================================== \r\n")  
 j += 1  
 break  
 if (i[0] >= 341) & (i[0] <= 371):  
 file.write("Question"+Question\_no[j] + " : Agree \r\n====================================================================== \r\n")  
 j += 1  
 break  
 if (i[0] >= 372) & (i[0] <= 402):  
 file.write("Question"+Question\_no[j] + " : Neutral \r\n====================================================================== \r\n")  
 j += 1  
 break  
 if (i[0] >= 403) & (i[0] <= 433):  
 file.write("Question"+Question\_no[j] + " : Disagree \r\n====================================================================== \r\n")  
 j += 1  
 break  
 if (i[0] >= 434) & (i[0] <= 474):  
 file.write("Question"+Question\_no[j] + " : Strongly Disagree \r\n====================================================================== \r\n")  
 j += 1  
 break  
file.close()  
cv2.waitKey(0)  
cv2.destroyAllWindows()

# Steps to reach the output :

1. After reading the input image , first resizing the image to make sure that any image has the same size so as the algorithm will work on any image size
2. Then checking if the image is rotated and detect the angle of rotation and the direction of rotation by calculating the slope
3. After rotating the image , thresholding is applied to remove any un-wanted data and focus on the needed features only
4. Finally assigning ranges for each question on the X-AXIS and Y-AXIS to be able to know the answer

# Test Cases 1 :

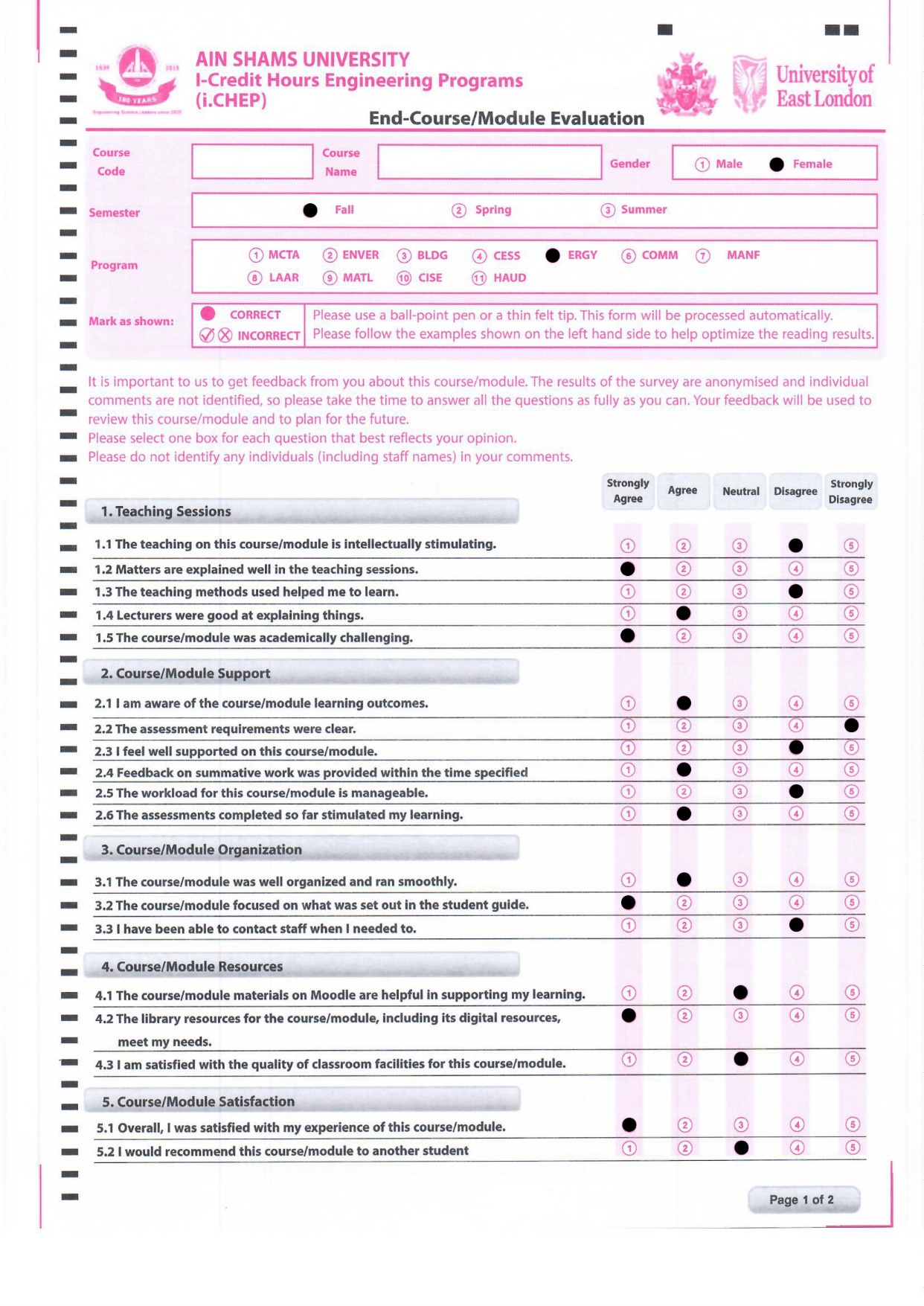


Figure 1 Test Sample 1

# Output for testcase 1:

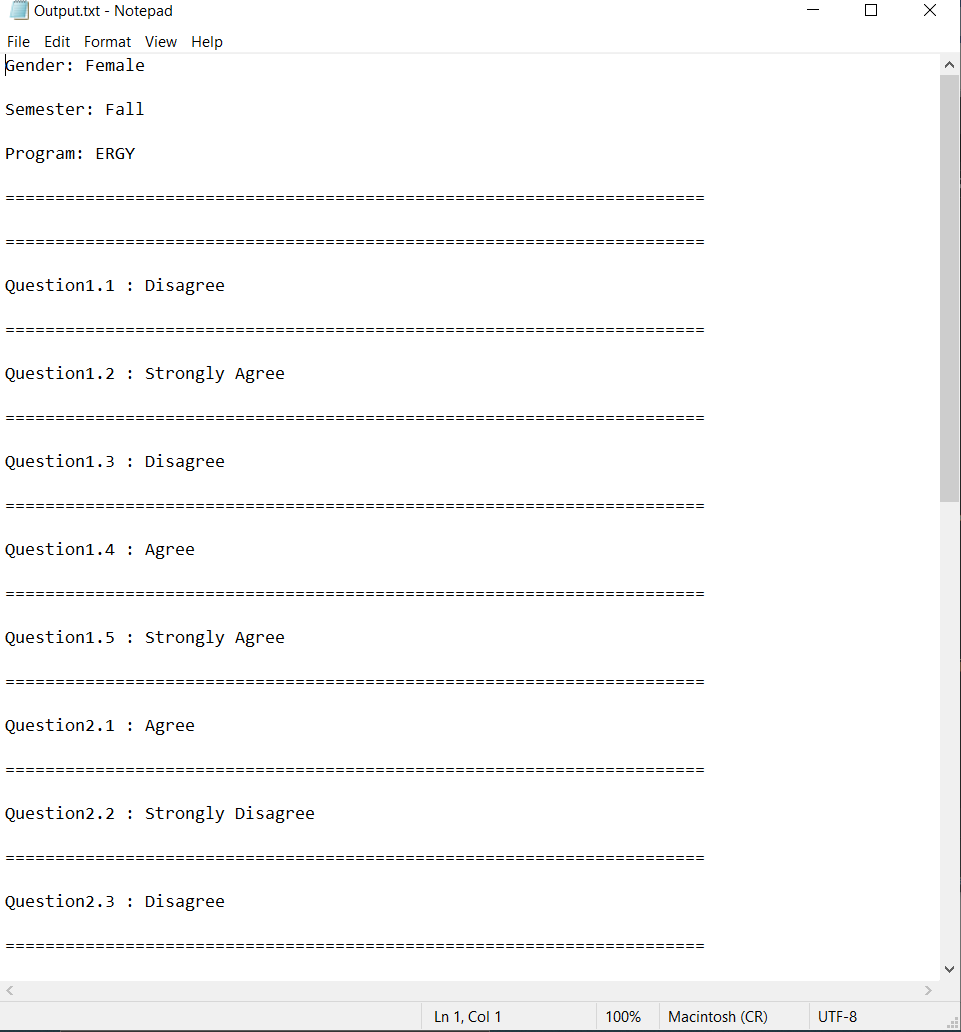


Figure 2 Test Case 1 Output File

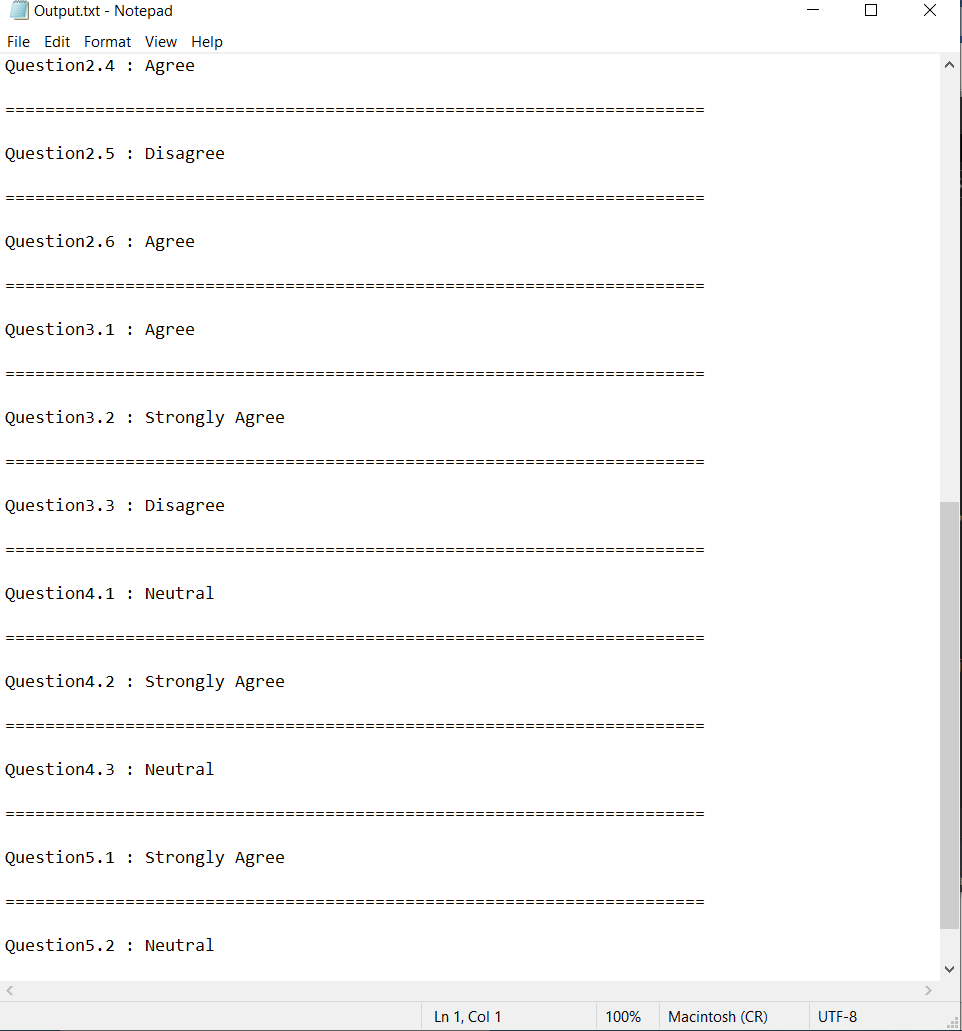


Figure 3 Test Case 1 Output File (Cont.)

# Test Case 2 :

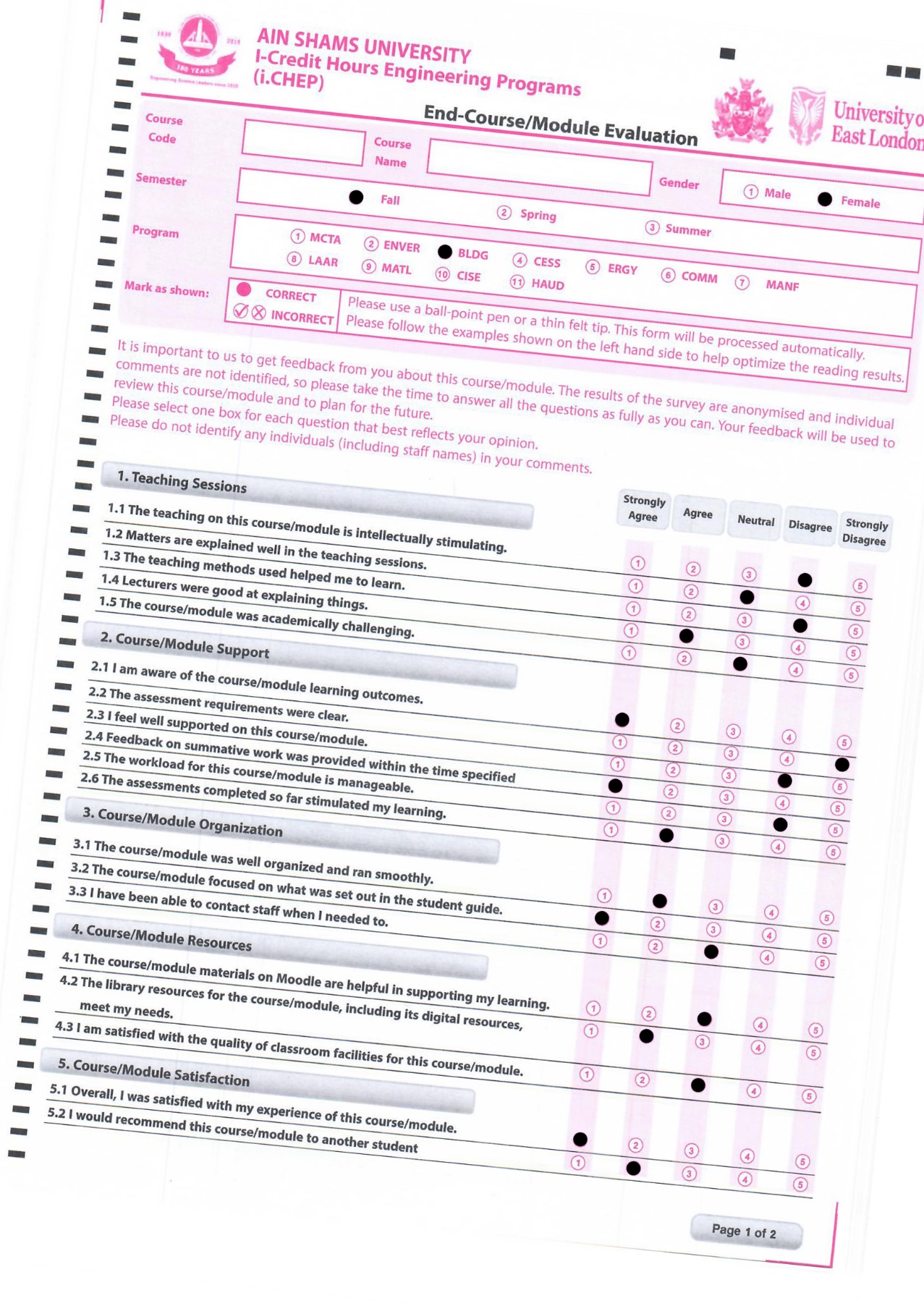


Figure 4 Test Sample 2

# Test Case 2 Output :

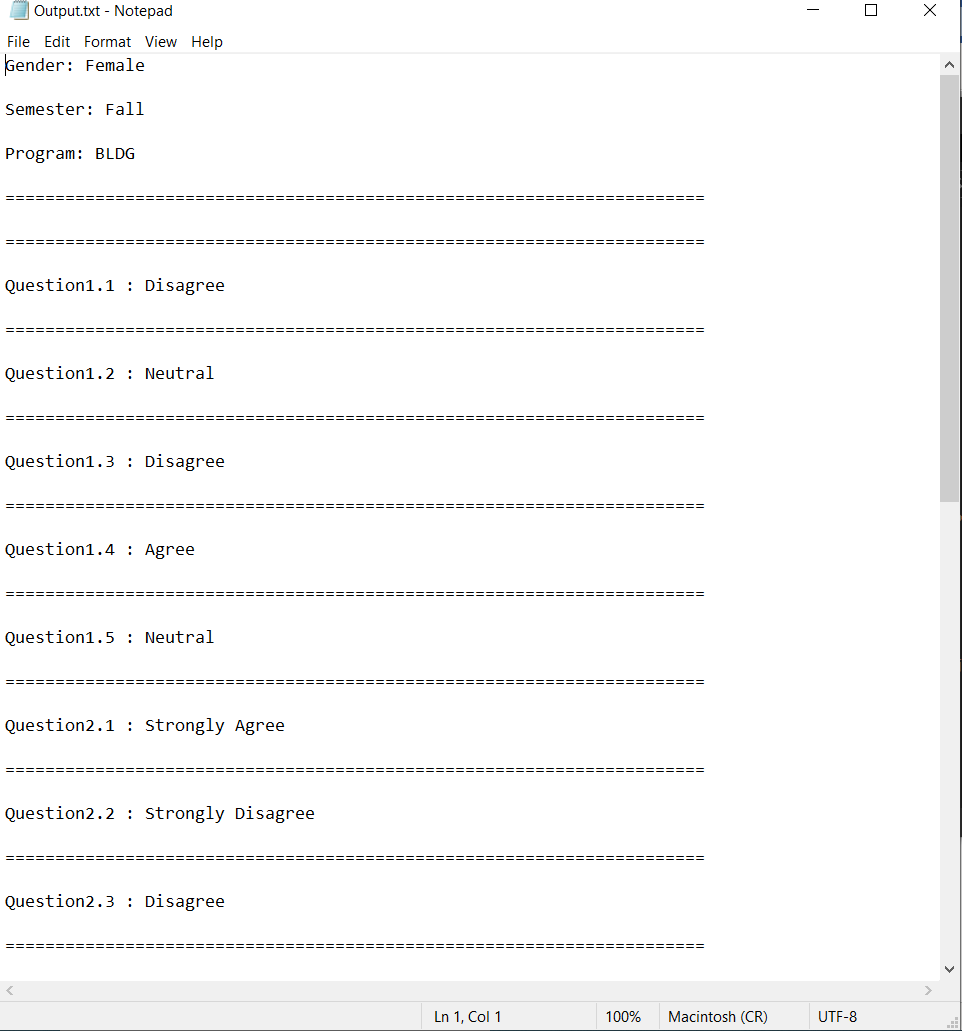


Figure 5 Test Case 2 Output File

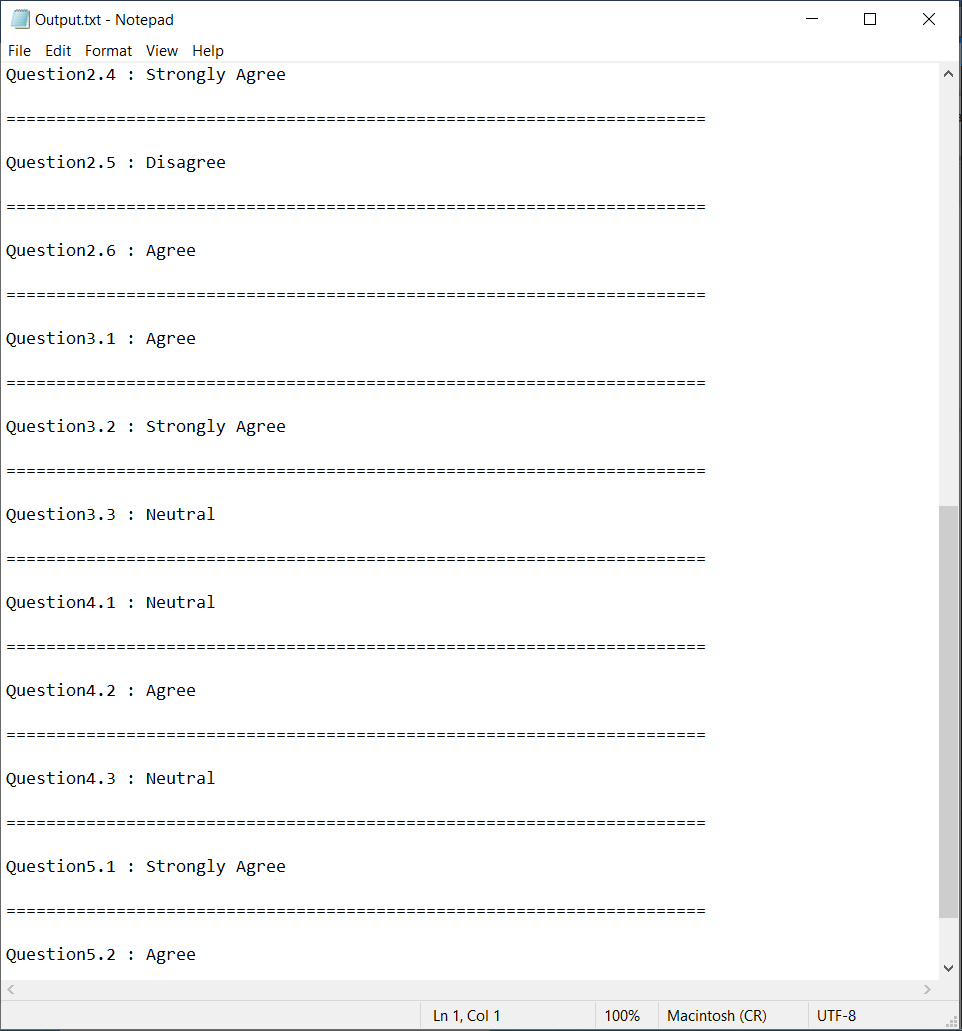


Figure 6 Test Case 2 Output File (Cont.)

# Test Case 3 :

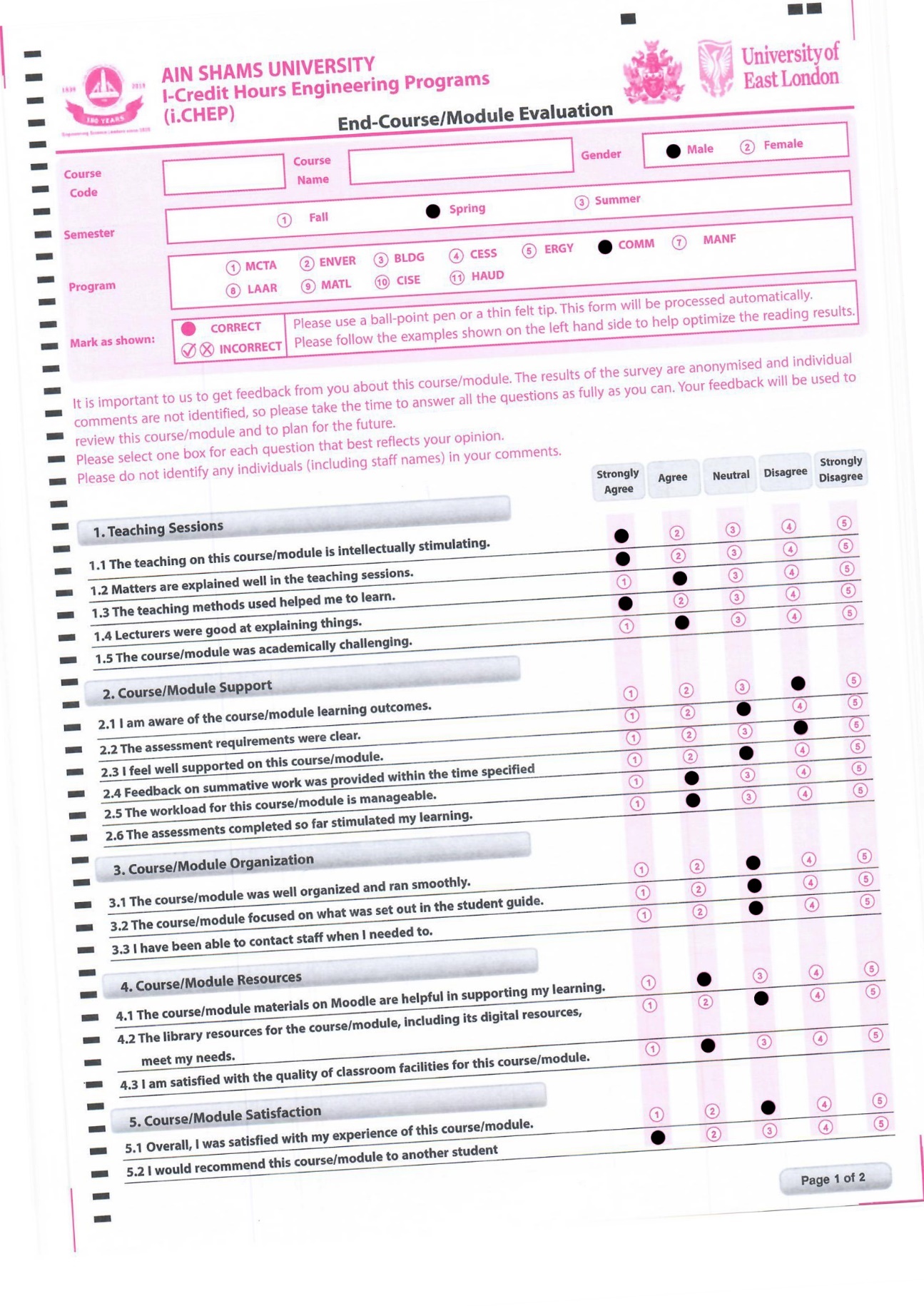


Figure 7 Test Sample 3

# Test Case 3 Output :

A screenshot of a social media post

Description automatically generated

Figure 8 Test Case 3 Output File

A screenshot of a computer

Description automatically generated

Figure 9 Test Case 3 Output File (Cont.)