**Functions Spot Check Electronic Answer Document (EAD)**

Use the following document to record your answers to the assignment spot check. You should then submit the completed EAD to the link provided on Moodle by your teacher.

|  |
| --- |
| **Question 1 part (a)** |
| **#Euan McElhoney**  **#09/12/14**  **#Functions spot check - 1**  **def password\_checker():**  **password = password\_input()**  **password\_validation(password)**  **def password\_input():**  **password = input("Please enter a password: ")**  **passwordlength = len(password)**  **return passwordlength**  **def password\_validation(passwordlength):**  **if passwordlength > 16:**  **print("Password to long")**  **password\_checker()**  **elif passwordlength < 8:**  **print("password too short")**  **password\_checker()**  **else:**  **print("password accepted")**  **password\_checker()** |
| **Question 1 part (b)** |
|  |

|  |
| --- |
| **Question 2 part (b)** |
| The program tries to display a negative amount of the grid which results in it displaying nothing |

|  |
| --- |
| **Question 2 part (c)** |
| # define functions  def GetGridRow(aRowLength):  # draws a single row using |\_ for each square  thisRow = '|\_' \* (aRowLength)  # add closing | to row  thisRow = thisRow + '|'  return thisRow  def DisplayGrid(aGridSize, aRow):  # display top of grid using \_ as top of each square  print(' \_' \* aGridSize)  # display rows of |\_| for each row  for rowCount in range(aGridSize):  print(aRow)  def GetGridSize():  validGrid = False  while validGrid is False:  thisGridSize = int(input("Please enter the size of the grid (max 20)"))  if thisGridSize <=20 and thisGridSize >0:  validGrid = True    return thisGridSize  # main program  def main():  thisGridSize = GetGridSize()  rowToDraw = GetGridRow(thisGridSize)  DisplayGrid(thisGridSize, rowToDraw)  main() |
| **Question 2 part (d)** |
|  |
| **Question 2 part (e)** |
|  |
| **Question 2 part (f)** |
|  |

|  |
| --- |
| **Question 3 part (a)** |
| #Euan McElhoney  #09/12/14  #Functions Spot Check - Q3  def journey\_cost\_calculator():  distance, mpg, fuelprice = GetInput()  total\_cost\_pound = CalculateCost(distance, mpg, fuelprice)  DisplayCost(total\_cost\_pound)    def GetInput():  distance = float(input("Please enter the distance in miles: "))  mpg = float(input("Please enter the Miles per Gallon of the veichle: "))  fuelprice = float(input("Please enter the cost of fuel per litre in pence: "))  return distance, mpg, fuelprice  def CalculateCost(distance, mpg, fuelprice):  gallon = distance / mpg  total\_litre = gallon \* 4.55  total\_cost\_pence = total\_litre \* fuelprice  total\_cost\_pound = total\_cost\_pence / 100  return total\_cost\_pound  def DisplayCost(total\_cost\_pound):  print("The total of your journey is £{0:.2f}".format(total\_cost\_pound))    journey\_cost\_calculator() |
| **Question 3 part (b)** |
|  |