Screenshots or program listings must be copied into appropriate cells in the following table.

Save this evidence document as **evidence\_** followed by your **centre number\_** **candidate number**, for example, **evidence\_ zz999\_9999** and insert your name, centre number and candidate number into the header above.

Examiners must be able to read the contents including any screenshots without the use of a magnifying glass. Answers that are not readable or are missing will not be awarded any marks.

Save this evidence document at regular intervals, for example every 10 minutes.

|  |
| --- |
| **Question 1** |
| **Part 1(a)(i)** |
| DataArray = [] #To store 25 integers |

|  |
| --- |
| **Part 1(a)(ii)** |
| def main():  RDataArray = open("Data.txt", "r")  for i in RDataArray:  DataArray.append(i) |

|  |
| --- |
| **Part 1(b)(i)** |
| def PrintArray(array):  for i in array:  print(i, end=" ") |

|  |
| --- |
| **Part 1(b)(ii)** |
| def main():  RDataArray = open("Data.txt", "r")  for i in RDataArray:  DataArray.append(i)  RDataArray.close()  PrintArray(DataArray) |

|  |
| --- |
| **Part 1(b)(iii)** |
|  |

|  |
| --- |
| **Part 1(c)** |
| def LinearSearch(array, x):  count = 0  for i in range(len(array)):  if array[i] == x:  count += 1  return count |

|  |
| --- |
| **Part 1(d)(i)** |
| def main():  UserInput = int(input("Please enter an integer between 0 and 100 inclusive:"))  while UserInput <0 or UserInput >100:  UserInput = int(input("Please enter an integer between 0 and 100 inclusive:"))  LinearSearch(DataArray, UserInput)  print("The number", UserInput, "is found", count, "times")    RDataArray = open("Data.txt", "r")  for i in RDataArray:  DataArray.append(i)  RDataArray.close()  PrintArray(DataArray) |

|  |
| --- |
| **Part 1(d)(ii)** |
|  |

|  |
| --- |
| **Question 2** |
| **Part 2(a)(i)** |
| class Vehicles:  #DECLARE ID : STRING  #DECLARE MaxSpeed : INTEGER  #DECLARE CurrentSpeed : INTEGER  #DECLARE IncreaseAmount : INTEGER  #DECLARE HorizontalPosition : INTEGER  def \_\_init\_\_(self, ID, MaxSpeed, IncreaseAmount, CurrentSpeed = 0, HorizontalPosition = 0):  self.\_\_ID = ID  self.\_\_MaxSpeed = MaxSpeed  self.\_\_IncreaseAmount = IncreaseAmount  self.\_\_CurrentSpeed = CurrentSpeed  self.\_\_HorizontalPosition = HorizontalPosition |

|  |
| --- |
| **Part 2(a)(ii)** |
| def GetCurrentSpeed(self):  return self.\_\_CurrentSpeed    def GetIncreaseAmount(self):  return self.\_\_IncreaseAmount    def GetHorizontalPosition(self):  return self.\_\_HorizontalPosition    def GetMaxSpeed(self):  return self.\_\_MaxSpeed |

|  |
| --- |
| **Part 2(a)(iii)** |
| def SetCurrentSpeed(self, CurrentSpeed):  self.\_\_CurrentSpeed = CurrentSpeed    def SetHorizontalPosition(self, HorizontalPosition):  self.\_\_HorizontalPosition = HorizontalPosition |

|  |
| --- |
| **Part 2(a)(iv)** |
| def IncreaseSpeed(self):  self.\_\_CurrentSpeed += self.\_\_IncreaseAmount  self.\_\_HorizontalPosition += self.\_\_CurrentSpeed  if self.\_\_CurrentSpeed > self.\_\_MaxSpeed:  print("Invalid, exceeds max speed of the vehicle.") |

|  |
| --- |
| **Part 2(b)(i)** |
| class Helicopter(Vehicle):  def \_\_init\_\_(self, ID, MaxSpeed, IncreaseAmount, MaxHeight, VerticalChange = 0):  super().\_\_init\_\_(ID, MaxSpeed, IncreaseAmount, CurrentSpeed, HorizontalPosition)  self.\_\_MaxHeight = MaxHeight  self.\_\_VerticalChange = VerticalChange  self.\_\_VerticalPostion = |

|  |
| --- |
| **Part 2(b)(ii)** |
| def IncreseSpeed(self):  global VerticalPosition, HorizontalPosition  self.\_\_VerticalPostion += self.\_\_VerticalChange  self.\_\_CurrentSpeed += self.\_\_IncreaseAmount  self.\_\_HorizontalPosition += self.\_\_CurrentSpeed  if self.\_\_VerticalPostion > self.\_\_MaxHeight:  print("Invalid, exceeds max height of the helicopter.")  if self.\_\_CurrentSpeed > self.\_\_MaxSpeed:  print("Invalid, exceeds max speed of the helicopter.") |

|  |
| --- |
| **Part 2(c)** |
| def Output(self):  global VerticalPosition  HorizontalPosition = self.\_\_HorizontalPosition  CurrentSpeed = self.\_\_CurrentSpeed  VerticalPosition = self.\_\_VerticalPostion  OutputHorizontal = print("The current horizontal position of the vehicle is", HorizontalPosition)  OutputCurrentSpeed = print("The current speed of the vehicle is", CurrentSpeed)  OutputVertical = print("The current vertical position of the vehicle is", VerticalPosition)  return OutputHorizontal, OutputVertical, OutputCurrentSpeed |

|  |
| --- |
| **Part 2(d)(i)** |
| def main():  NewCar = Vehicles("Tiger", 100, 20)  NewCar.IncreaseSpeed()  NewCar.IncreaseSpeed()  NewCar.Output()    NewHelicopter = Helicopter("Lion", 350, 40, 100, 3)  NewHelicopter.IncreaseSpeed()  NewHelicopter.IncreaseSpeed() |

|  |
| --- |
| **Part 2(d)(ii)** |
| *{Copy and paste screenshot for Question 2(d)(ii) here.* |

|  |
| --- |
| **Question 3** |
| **Part 3(a)** |
| global Animal, Colour  Animal = [] #Contain 20 elements  Colour = [] #Contain 10 elements  AnimalTopPointer = 0  ColourTopPointer = 0 |

|  |
| --- |
| **Part 3(b)(i)** |
| def PushAnimal(DTP):  global AnimalTopPointer, ColourTopPointer  if AnimalTopPointer == 20:  return False  else:  Animal.append(DTP)  AnimalTopPointer += 1  return True |

|  |
| --- |
| **Part 3(b)(ii)** |
| def PopAnimal():  global AnimalTopPointer, ReturnData  ReturnData = ""  if AnimalTopPointer == 0:  return ""  else:  ReturnData = Animal[AnimalTopPointer - 1]  AnimalTopPointer -= 1  return ReturnData |

|  |
| --- |
| **Part 3(b)(iii)** |
| def ReadData():  global AnimalTopPointer, ColourTopPointer  RAnimals = open("AnimalData.txt", "r")  for thisline in RAnimals:  PushAnimal(thisline)  RAnimals.close() |

|  |
| --- |
| **Part 3(b)(iv)** |
| def PushColour(CTP):  global ColourTopPointer  if ColourTopPointer == 10:  return False  else:  Colour[ColourTopPointer] = CTP  ColourTopPointer += 1  return True    def PopColour():  global ColourTopPointer, ReturnData1  ReturnData1 = 0  if ColourTopPointer == 0:  return ""  else:  ReturnData1 = Colour[ColourTopPointer - 1]  ColourTopPointer -= 1  return ReturnData1 |

|  |
| --- |
| **Part 3(b)(v)** |
| def ReadData():  try:  RAnimals = open("AnimalData.txt", "r")  for line in RAnimals:  PushAnimal(line.strip())  RAnimals.close()  except IOError:  print("Please check if the file exists")  try:  RColours = open("ColourData.txt", "r")  for line1 in RColours:  PushColour(line1.strip())  RColours.close()  except IOError:  print("Please check if the file exists") |

|  |
| --- |
| **Part 3(c)** |
| def OutputItem():  global AnimalTopPointer, ColourTopPointer  CurrentColour = PopColour(Colour)  CurrentAnimal = PopAnimal(Animal)  if CurrentColour == "":  print("No colour")  PushAnimal(CurrentAnimal)  else:  if CurrentAnimal == "":  print("No animal")  PushColour(CurrentColour)  else:  print(CurrentColour + " " + CurrentAnimal) |

|  |
| --- |
| **Part 3(d)(i)** |
| def MainProgram():  global AnimalTopPointer, ColourTopPointer  ReadData1()  for x in range(4):  OutputItem() |

|  |
| --- |
| **Part 3(d)(ii)** |
|  |