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 missing will not be awarded any marks.

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

|  |
| --- |
| **Question 1** |
| **Part 1(a)** |
| *TheData = [20, 3, 4, 8, 12, 99, 4, 26, 4]* |
| **Part 1(b)** |
| *def InsertionSort(TheData):*  *for count in range(len(TheData)):*  *DataToInsert = TheData[count]*  *Inserted = 0*  *NextValue = count - 1*  *while NextValue >= 0 and Inserted != 1:*  *if DataToInsert < TheData[NextValue]:*  *TheData[NextValue + 1] = TheData[NextValue]*  *NextValue -= 1*  *TheData[NextValue + 1] = DataToInsert*  *else:*  *Inserted = 1* |
| **Part 1(c)** |
| *def OutputArray(arr):*  *for i in arr:*  *print(i, end = " ")* |
| **Part 1(d)(i)** |
| *def main():*  *TheData = [20, 3, 4, 8, 12, 99, 4, 26, 4]*  *print("Before")*  *OutputArray(TheData)*  *print("\nAfter")*  *InsertionSort(TheData)*  *OutputArray(TheData)*  *if \_\_name\_\_ == "\_\_main\_\_":*  *main()* |
| **Part 1(d)(ii)** |
|  |
| **Part 1(e)(i)** |
| *def Find(arr, n):*  *for i in arr:*  *if i == n:*  *print("found")*  *return True*  *print("not found")*  *return False* |
| **Part 1(e)(ii)** |
|  |
| **Question 2** |
| **Part 2(a)** |
| *class HiddenBox:*  *# \_\_BoxName string*  *# \_\_Creator string*  *# \_\_DateHidden date*  *# \_\_GameLocation string*  *# \_\_LastFinds 2D array*  *# \_\_Active boolean* |
| **Part 2(b)** |
| *def \_\_init\_\_(self, name, creator, date, location):*  *self.\_\_BoxName = name*  *self.\_\_Creator = creator*  *self.\_\_DateHidden = date*  *self.\_\_GameLocation = location*  *self.\_\_Active = False*  *self.\_\_LastFinds = [["" for i in range(2)] for j in range(10)]* |
| **Part 2(c)** |
| *def GetBoxName(self):*  *return self.\_\_BoxName*  *def GetGameLocation(self):*  *return self.\_\_GameLocation* |
| **Part 2(d)(i)** |
| *def main():*  *TheBoxes = []*  *if \_\_name\_\_ == "\_\_main\_\_":*  *main()* |
| **Part 2(d)(ii)** |
| *def NewBox(arr):*  *name = input("Enter the name of the box: ")*  *creator = input("Enter the creator: ")*  *day = int(input("Enter the date day: "))*  *month = int(input("Enter the date month: "))*  *year = int(input("Enter the date yaer: "))*  *d = date(year, month, day)*  *location = input("Enter location: ")*  *arr.append(HiddenBox(name, creator, d, location))* |
| **Part 2(d)(iii)** |
| *def main():*  *TheBoxes = []*  *NewBox(TheBoxes)*  *if \_\_name\_\_ == "\_\_main\_\_":*  *main()* |
| **Part 2(e)** |
| *class PuzzleBox(HiddenBox):*  *def \_\_init\_\_(self, name, creator, date, location, puzzle, solution):*  *super().\_\_init\_\_(name, creator, date, location)*  *self.\_\_PuzzleText = puzzle*  *self.\_\_Solution = solution* |
| **Question 3** |
| **Part 3(a)** |
| *global QueueData, start, end*  *QueueData = [""] \* 20*  *start = 0*  *end = -1* |
| **Part 3(b)** |
| *def Enqueue(item):*  *global QueueData, start, end*  *end += 1*  *if end >= 20:*  *return False*  *else:*  *QueueData[end] = item*  *return True* |
| **Part 3(c)** |
| *def ReadFile():*  *file = input("Enter file name: ")*  *try:*  *with open(file, "r", encoding = "utf-8") as f:*  *for line in f:*  *if not Enqueue(line.strip()): return 1*  *return 2*  *except IOError:*  *return -1* |
| **Part 3(d)(i)** |
| *def main():*  *c = ReadFile()*  *if c == 2:*  *print("Successfully added!")*  *elif c == 1:*  *print("Partially added as queue is full.")*  *elif c == -1:*  *print("File does not exist!")*  *if \_\_name\_\_ == "\_\_main\_\_":*  *main()* |
| **Part 3(d)(ii)** |
|  |
| **Part 3(e)** |
| *def Remove():*  *global QueueData, start, end*  *res = ""*  *for i in range(2):*  *if end == start:*  *return "No Items"*  *else:*  *res += QueueData[start] + " "*  *start += 1*  *return res[:len(res) - 1]* |