Candidate name: ...Chan Zun Mun Terence...............................................

Centre number: ...3011...............................................

Index number: .....1630.............................................

Programming language used: .....Python................................................................

|  |
| --- |
| **Question 2** |
| **Evidence 4**  *Program code*  ###Task 2.1##################################  ###From SCORES.TXT  sampleData = [72,45,120,57,76,29,40,42,113,77,64,29,91,122,134,139,24,92,136,73,78,62,27,126,76,116,67,34,20,90,75,128,128,89,98,41,37,58,124,24,101,132]  ###From QUICKSORT.TXT  def QuickSort(Scores):  QuickSortHelper(Scores, 0, len(Scores) - 1)  return Scores  def QuickSortHelper(Scores, First, Last):  if First < Last:  SplitPoint = Partition(Scores, First, Last)  QuickSortHelper(Scores, First, SplitPoint - 1)  QuickSortHelper(Scores, SplitPoint + 1, Last)  return Scores  def Partition(Scores, First, Last):  PivotValue = Scores[First]  LeftMark = First + 1  RightMark = Last  Done = False  while (Done == False):  while LeftMark <= RightMark and Scores[LeftMark] <= PivotValue:  LeftMark = LeftMark + 1  while Scores[RightMark] >= PivotValue and RightMark >= LeftMark:  RightMark = RightMark - 1  if RightMark < LeftMark:  Done = True  else:  Temp = Scores[LeftMark]  Scores[LeftMark] = Scores[RightMark]  Scores[RightMark] = Temp  #<swap Scores[First] with Scores[RightMark]>  temp = Scores[First]  Scores[First] = Scores[RightMark]  Scores[RightMark] = temp  return RightMark |
| **Evidence 5**  *Program code*  ###Task 2.2################################  def output():  print(sampleData)  print("Data before Sorting:")  output()  QuickSort(sampleData)  print("Data after Sorting:")  output() |
| **Evidence 6**  *Screenshot* |
| **Question 3** |
| **Evidence 7**  *Program code*  ###Task 3.1  class Record:  def \_\_init\_\_(self, name, number):  self.name = name  self.number = number #Telephone Number  ###Getter###  def getName(self):  return self.name  def getNumber(self):  return self.number  ###Setter###  def setName(self, name):  self.name = name  def setNumber(self, number):  self.number = number  class RecordsStorage:  def \_\_init\_\_(self, no = 500):  self.records = []  for i in range(no): #Store Number of records  record = Record("", None)  self.records.append(record)  def length(self):  return len(self.records)  def access(self, pos):  return self.records[pos] |
| **Evidence 8**  *Program code*  ###Task 3.2  recordStorage = RecordsStorage()  def extractDataFromFile():  dataFile = open("HASHEDDATA.TXT")  lines = dataFile.read().split("\n")  dataFile.close()  for line in lines:  ###Getting Data from line  lineData = line.split(",")  index = int(lineData[0])  name = lineData[1]  number = lineData[2]  ###Storing Values in Data Structure  record = recordStorage.access(index)  record.setName(name)  record.setNumber(number)    def DisplayValues():  ###Table Heading  print(f"{'Index':<5} | {'PersonsName':<20} | {'TelephoneNumber':<8}")  ###Loop through data structure  for i in range(recordStorage.length()):  record = recordStorage.access(i)  if record.getName() != "": #If name is present  print(f"{i:<5} | {record.getName():<20} | {record.getNumber():<8}")  extractDataFromFile()  DisplayValues()  print() |
| **Evidence 9**  *Screenshot* |
| **Evidence 10**  *Program code*  ###Task 3.3  def GenerateHash(SearchName):  HashTotal = 0  ###Loop through position of character in array  for char in range(len(SearchName)):  asciiCode = ord(SearchName[char])  HashTotal += asciiCode \* (char + 1) #Position to multiply starts from 1  Hash = HashTotal % 500  return Hash  print("Task 3.3:")  print("Hash value of Tait Davinder is :" , GenerateHash("Tait Davinder"))  print("Hash value of Anandan Yeo is :" , GenerateHash("Anandan Yeo"))  print() |
| **Evidence 11**  *Screenshot* |
| **Evidence 12**  *Program code*  ###Task 3.4  def search(SearchName):  index = GenerateHash(SearchName)  record = recordStorage.access(index)  while record.getName() != SearchName and\  record.getName() != "": #Name not Empty  index += 1  record = recordStorage.access(index)    if recordStorage.access(index).getName() == "": #If Name Empty  print("NOT FOUND")  else:  print(f"{'Index':<5} | {'PersonsName':<20} | {'TelephoneNumber':<8}")  print(f"{index:<5} | {record.getName():<20} | {record.getNumber():<8}")  #Searching  print("Task 3.4:")  print("Search 1:Charlie Love")  search("Charlie Love")  print()  print("Search 2:Chin Tan")  search("Chin Tan")  print()  print("Search 3:John Barrowman")  search("John Barrowman")  print() |
| **Evidence 13**  *Screenshot* |