|  |
| --- |
| Evidence 1 |
| **class** **BookRec**:  **def** \_\_init\_\_(self, BookID, Title, Pointer):  self.\_\_BookID = BookID  self.\_\_Title = Title  self.\_\_Pointer = Pointer  **def** GetBookID(self):  **return** self.\_\_BookID  **def** GetTitle(self):  **return** self.\_\_Title  **def** GetPointer(self):  **return** self.\_\_Pointer  **def** SetBookID(self, BookID):  self.\_\_BookID = BookID  **def** SetTitle(self, Title):  self.\_\_Title = Title  **def** SetPointer(self, Pointer):  self.\_\_Pointer = Pointer |
| Evidence 2 |
| **class** **LinkedList**: *# dynamic linked list*  **def** \_\_init\_\_(self):  self.\_\_Start = None *# null*    **def** IsEmpty(self):  **return** (self.\_\_Start == None)  **def** DisplayLinkedList(self):  curr = self.\_\_Start *# current node*  **if** curr == None:  *#empty linked list*  **print**("Empty linked list")  **else**:  **print**()  **print**("-" \* 52)  **print**("| {0:^15} | {1:^30} |".format("BookID", "Title"))  **print**("-" \* 52)  **while** curr != None:  **print**("| {0:^15} | {1:^30} |".format(curr.GetBookID(), curr.GetTitle()))  curr = curr.GetPointer()  **print**("-" \* 52)    **def** AddNode(self, BookID, Title):  NewNode = BookRec(BookID, Title, None)  **if** self.IsEmpty():  self.\_\_Start = NewNode  **else**:  curr = self.\_\_Start  **while** curr.GetPointer() != None:  curr = curr.GetPointer()  curr.SetPointer(NewNode)  **def** SearchNode(self, BookID):  curr = self.\_\_Start  **while** curr != None **and** curr.GetBookID() != BookID:  curr = curr.GetPointer()  **if** curr == None:  **return** False *# is within linked list*  **else**:  **return** True *# is not in linked list*  *# return (curr == None)*  **def** DeleteNode(self, BookID):  **if** self.SearchNode(BookID):  prev = self.\_\_Start  curr = self.\_\_Start  **while** curr != None **and** curr.GetBookID() != BookID:  prev = curr  curr = curr.GetPointer()  prev.SetPointer(curr.GetPointer()) |
| Evidence 3 |
| **class** **HashTable**:  **def** \_\_init\_\_(self, Size = 17): *# Initialise()*  self.\_\_Size = Size  self.\_\_Slots = [None] + [LinkedList() **for** i **in** range(Size)]  *# array of empty linked lists.*  *# self.\_\_Slots[0] = None as hash table is 1-based.*  **def** Hash(self, BookID):  ASCIISum = 0  **for** character **in** BookID:  ASCIISum += ord(character) *# add ASCII value to ASCII sum*  Address = (ASCIISum % self.\_\_Size) + 1 *# address = remainder + 1*  **return** Address  **def** Display(self):  **for** i **in** range(self.\_\_Size):  Address = i + 1  CurrentLinkedList = self.\_\_Slots[Address]  **print**("[{0}] in hash table: ".format(Address), end='')  CurrentLinkedList.DisplayLinkedList()  **print**()  **def** Put(self, BookID, Title):  Address = self.Hash(BookID)  self.\_\_Slots[Address].AddNode(BookID, Title)  **def** Remove(self, BookID):  Address = self.Hash(BookID)  CurrentLinkedList = self.\_\_Slots[Address]  **if** CurrentLinkedList.SearchNode(BookID):  CurrentLinkedList.DeleteNode(BookID)  **print**("{0} has been removed from the hash table.".format(BookID))  **else**:  **print**("{0} cannot be removed as it is not found in the hash table.".format(BookID))    **def** Search(self, BookID):  Address = self.Hash(BookID)  CurrentLinkedList = self.\_\_Slots[Address]  **if** CurrentLinkedList.SearchNode(BookID):  **return** True  **else**:  **return** False  *# alternatively: return CurrentLinkedList.SearchNode(BookID)* |
| Evidence 4 |
| **Screenshots**      **Program code**  Books = HashTable(17)  Books.Put("CS733", "Basic algorithms")  Books.Put("AB944", "Master Computing")  Books.Put("KS293", "Data structures")  Books.Put("BK232", "Programming exercises")  Books.Put("PK199", "Testing Python")  Books.Display()  Books.Remove("AB944")  Books.Display()  **print**(Books.Search("PK199")) |