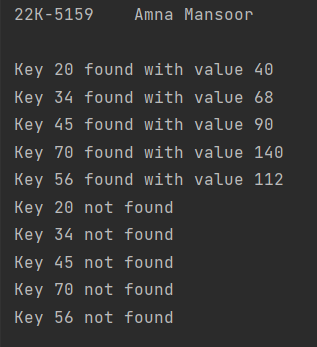
**22K-5159** | **Amna Mansoor** | **LAB-11** | **BSE-3B**

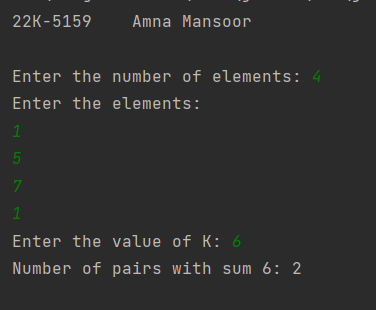
*Task 1:*

public class Task1 {  
 public static void main(String[] args) {  
 System.*out*.println("22K-5159 Amna Mansoor\n");  
 int[] keys = {20, 34, 45, 70, 56};  
 HashTable hashTable = new HashTable(10);  
  
 // inserting elements into the table  
 for (int key : keys) {  
 hashTable.insert(key, key \* 2);  
 }  
  
 // searching and displaying elements  
 for (int key : keys) {  
 Node result = hashTable.search(key);  
 if (result != null) {  
 System.*out*.println("Key " + key + " found with value " + result.value);  
 } else {  
 System.*out*.println("Key " + key + " not found");  
 }  
 }  
  
 // deleting elements  
 for (int key : keys) {  
 hashTable.delete(key);  
 }  
  
 // searching after deletion  
 for (int key : keys) {  
 Node result = hashTable.search(key);  
 if (result != null) {  
 System.*out*.println("Key " + key + " found with value " + result.value);  
 } else {  
 System.*out*.println("Key " + key + " not found");  
 }  
 }  
 }  
}  
class Node {  
 int key;  
 int value;  
 Node next;  
  
 public Node(int key, int value) {  
 this.key = key;  
 this.value = value;  
 this.next = null;  
 }  
}  
  
class LinkedList {  
 Node head;  
  
 public void insert(int key, int value) {  
 Node newNode = new Node(key, value);  
 newNode.next = head;  
 head = newNode;  
 }  
  
 public Node search(int key) {  
 Node current = head;  
 while (current != null) {  
 if (current.key == key) {  
 return current;  
 }  
 current = current.next;  
 }  
 return null;  
 }  
  
 public void delete(int key) {  
 Node current = head;  
 Node prev = null;  
  
 while (current != null && current.key != key) {  
 prev = current;  
 current = current.next;  
 }  
  
 if (current != null) {  
 if (prev != null) {  
 prev.next = current.next;  
 } else {  
 head = current.next;  
 }  
 }  
 }  
}  
  
class HashTable {  
 private LinkedList[] table;  
 private int size;  
  
 public HashTable(int size) {  
 this.size = size;  
 table = new LinkedList[size];  
 for (int i = 0; i < size; i++) {  
 table[i] = new LinkedList();  
 }  
 }  
  
 private int hash(int key) {  
 return key % size;  
 }  
  
 public void insert(int key, int value) {  
 int index = hash(key);  
 table[index].insert(key, value);  
 }  
  
 public Node search(int key) {  
 int index = hash(key);  
 return table[index].search(key);  
 }  
  
 public void delete(int key) {  
 int index = hash(key);  
 table[index].delete(key);  
 }  
}

**

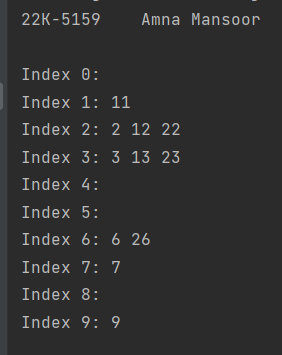
*Task 2:*

import java.util.Scanner;  
  
public class Task2 {  
 public static void main(String[] args) {  
 System.*out*.println("22K-5159 Amna Mansoor\n");  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Enter the number of elements: ");  
 int N = scanner.nextInt();  
  
 int[] arr = new int[N];  
 System.*out*.println("Enter the elements:");  
 for (int i = 0; i < N; i++) {  
 arr[i] = scanner.nextInt();  
 }  
  
 System.*out*.print("Enter the value of K: ");  
 int K = scanner.nextInt();  
  
 int result = *countPairs*(arr, K);  
 System.*out*.println("Number of pairs with sum " + K + ": " + result);  
 }  
 public static int countPairs(int[] arr, int K) {  
 int count = 0;  
 HashTable hashTable = new HashTable();  
  
 for (int i = 0; i < arr.length; i++) {  
 int complement = K - arr[i];  
 if (hashTable.search(complement)) {  
 count++;  
 }  
 hashTable.insert(arr[i]);  
 }  
  
 return count;  
 }  
  
  
}  
class HashTable {  
 private static final int *TABLE\_SIZE* = 100;  
 private Node[] table;  
  
 public HashTable() {  
 table = new Node[*TABLE\_SIZE*];  
 }  
  
 public void insert(int key) {  
 int index = hash(key);  
 Node newNode = new Node(key);  
 newNode.next = table[index];  
 table[index] = newNode;  
 }  
  
 public boolean search(int key) {  
 int index = hash(key);  
 Node current = table[index];  
  
 while (current != null) {  
 if (current.key == key) {  
 return true;  
 }  
 current = current.next;  
 }  
  
 return false;  
 }  
  
 private int hash(int key) {  
 return Math.*abs*(key % *TABLE\_SIZE*);  
 }  
}  
  
class Node {  
 int key;  
 Node next;  
  
 public Node(int key) {  
 this.key = key;  
 this.next = null;  
 }  
}

******

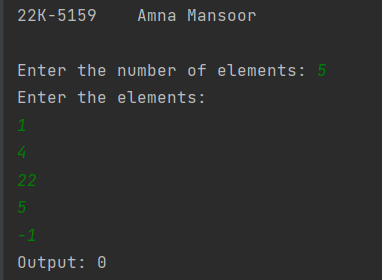
*Task 3:*

public class Task3 {  
 public static void main(String[] args) {  
 System.*out*.println("22K-5159 Amna Mansoor\n");  
 int[] values = {3, 2, 9, 6, 11, 13, 7, 12, 23, 22, 26};  
 int m = 10;  
  
 HashTable hashTable = new HashTable(m);  
  
 // inserting values into the hash table  
 for (int value : values) {  
 hashTable.insert(value);  
 }  
  
 // sorting each chain  
 hashTable.sortChains();  
  
 // displaying the sorted hash table  
 hashTable.displayTable();  
 }  
}  
  
class Node {  
 int value;  
 Node next;  
  
 public Node(int value) {  
 this.value = value;  
 this.next = null;  
 }  
}  
  
class LinkedList {  
 Node head;  
  
 public void insert(int value) {  
 Node newNode = new Node(value);  
 newNode.next = head;  
 head = newNode;  
 }  
  
 public void sort() {  
 if (head == null || head.next == null) {  
 return; // Already sorted or empty list  
 }  
  
 boolean swapped;  
 do {  
 swapped = false;  
 Node current = head;  
 Node next = head.next;  
  
 while (next != null) {  
 if (current.value > next.value) {  
 int temp = current.value;  
 current.value = next.value;  
 next.value = temp;  
 swapped = true;  
 }  
  
 current = next;  
 next = next.next;  
 }  
 } while (swapped);  
 }  
  
  
 public void display() {  
 Node current = head;  
 while (current != null) {  
 System.*out*.print(current.value + " ");  
 current = current.next;  
 }  
 System.*out*.println();  
 }  
}  
  
class HashTable {  
 private LinkedList[] table;  
 private int size;  
  
 public HashTable(int size) {  
 this.size = size;  
 table = new LinkedList[size];  
 for (int i = 0; i < size; i++) {  
 table[i] = new LinkedList();  
 }  
 }  
  
 private int hash(int value) {  
 return value % size;  
 }  
  
 public void insert(int value) {  
 int index = hash(value);  
 table[index].insert(value);  
 }  
  
 public void sortChains() {  
 for (int i = 0; i < size; i++) {  
 table[i].sort();  
 }  
 }  
  
 public void displayTable() {  
 for (int i = 0; i < size; i++) {  
 System.*out*.print("Index " + i + ": ");  
 table[i].display();  
 }  
 }  
}

******

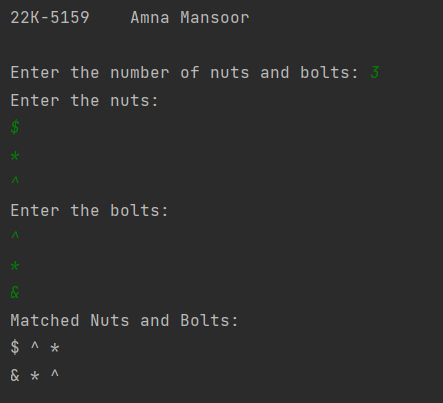
*Task 4:*

import java.util.Scanner;  
  
class HashTable {  
 private static final int *TABLE\_SIZE* = 100;  
 private Node[] table;  
  
 public HashTable() {  
 table = new Node[*TABLE\_SIZE*];  
 }  
  
 public void insert(int key) {  
 int index = hash(key);  
 Node newNode = new Node(key);  
 newNode.next = table[index];  
 table[index] = newNode;  
 }  
  
 public boolean search(int key) {  
 int index = hash(key);  
 Node current = table[index];  
  
 while (current != null) {  
 if (current.key == key) {  
 return true;  
 }  
 current = current.next;  
 }  
  
 return false;  
 }  
  
 private int hash(int key) {  
 return Math.*abs*(key % *TABLE\_SIZE*);  
 }  
}  
  
class Node {  
 int key;  
 Node next;  
  
 public Node(int key) {  
 this.key = key;  
 this.next = null;  
 }  
}  
  
public class Task4 {  
 public static int findTriplets(int[] arr) {  
 int n = arr.length;  
  
 for (int i = 0; i < n - 1; i++) {  
 HashTable hashTable = new HashTable();  
  
 for (int j = i + 1; j < n; j++) {  
 int complement = -(arr[i] + arr[j]);  
  
 if (hashTable.search(complement)) {  
 return 1; // Triplet found  
 }  
  
 hashTable.insert(arr[j]);  
 }  
 }  
  
 return 0; // Triplet not found  
 }  
  
 public static void main(String[] args) {  
 System.*out*.println("22K-5159 Amna Mansoor\n");  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Enter the number of elements: ");  
 int n = scanner.nextInt();  
  
 int[] arr = new int[n];  
 System.*out*.println("Enter the elements:");  
 for (int i = 0; i < n; i++) {  
 arr[i] = scanner.nextInt();  
 }  
  
 int result = *findTriplets*(arr);  
 System.*out*.println("Output: " + result);  
 }  
}

******

*Task 5:*

import java.util.Scanner;  
  
public class Task5 {  
 public static void main(String[] args) {  
 System.*out*.println("22K-5159 Amna Mansoor\n");  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Enter the number of nuts and bolts: ");  
 int n = scanner.nextInt();  
  
 char[] nuts = new char[n];  
 char[] bolts = new char[n];  
  
 System.*out*.println("Enter the nuts:");  
 for (int i = 0; i < n; i++) {  
 nuts[i] = scanner.next().charAt(0);  
 }  
  
 System.*out*.println("Enter the bolts:");  
 for (int i = 0; i < n; i++) {  
 bolts[i] = scanner.next().charAt(0);  
 }  
  
 NutBoltMatcher.*matchNutsAndBolts*(nuts, bolts, 0, nuts.length - 1);  
  
 System.*out*.println("Matched Nuts and Bolts:");  
 NutBoltMatcher.*printArray*(nuts);  
 NutBoltMatcher.*printArray*(bolts);  
 }  
}  
  
class NutBoltMatcher {  
 public static void matchNutsAndBolts(char[] nuts, char[] bolts, int low, int high) {  
 if (low < high) {  
 // choosing the last character of bolts array as the pivot  
 int pivotIndex = *partition*(nuts, low, high, bolts[high]);  
  
 // using the pivot to partition the bolts array  
 *partition*(bolts, low, high, nuts[pivotIndex]);  
  
 // recursively matching the remaining portions  
 *matchNutsAndBolts*(nuts, bolts, low, pivotIndex - 1);  
 *matchNutsAndBolts*(nuts, bolts, pivotIndex + 1, high);  
 }  
 }  
  
 private static int partition(char[] arr, int low, int high, char pivot) {  
 int i = low;  
 int j = low;  
  
 // moving the elements smaller than or equal to pivot to the left  
 for (; j < high; j++) {  
 if (arr[j] < pivot) {  
 *swap*(arr, i, j);  
 i++;  
 } else if (arr[j] == pivot) {  
 // finding the matching element and bring it to the end  
 *swap*(arr, j, high);  
 j--;  
 }  
 }  
 // moving the matching element to its correct position  
 *swap*(arr, i, high);  
 return i;  
 }  
  
 private static void swap(char[] arr, int i, int j) {  
 char temp = arr[i];  
 arr[i] = arr[j];  
 arr[j] = temp;  
 }  
  
 public static void printArray(char[] arr) {  
 for (char c : arr) {  
 System.*out*.print(c + " ");  
 }  
 System.*out*.println();  
 }  
}

******