Lab 1

```
package test;

          class SimpleArray{
                    private long[] a;
                     public SimpleArray(int size)
                     {a = new long[size];}

                    public void setElem(int index, long value)
                    {a[index] = value;}

                    public long getElem(int index)
                    {return a[index];}

                    public int findMax(int nElems) {
                              if (nElems <+ 0) {
                                        return -1;
                              }
                              int maxIndex = 0;
                              for (int i = 1; i < nElems;  i++) {
                                        if (a[i] > a[maxIndex]) {
                                                  maxIndex = i;
                                        }
                              }
                              return maxIndex;
                    }

                    public Long removeMax(int nElems) {
                              int maxIndex = findMax(nElems);
                              if(maxIndex == -1) {
                                        System.out.println("Cannot find max. Array is empty.");
                                        return null;
                              }
                              long maxValue = a[maxIndex];
                              for (int i = maxIndex; i < nElems -1; i++) {
                                        a[i] = a[i + 1];
                              }
                              System.out.println(maxValue + "has been removed.");
                              return maxValue;
                    }
          }

          class LabArray {
                    public static void main(String[] args) {
                              SimpleArray arr;
                              arr = new SimpleArray(100);
                              int nElems = 0;
                              int j;

                              arr.setElem(0, 77);
                              arr.setElem(1, 99);
                              arr.setElem(2, 44);
                              arr.setElem(3, 55);
                              arr.setElem(4, 22);
                              arr.setElem(5, 88);
                              arr.setElem(6, 11);
                              arr.setElem(7, 00);
                              arr.setElem(8, 66);
                              arr.setElem(9, 33);
                              nElems = 10;

                              for (j = 0; j < nElems; j++)
                                        System.out.print(arr.getElem(j) + " ");
                     System.out.println("");

                              int maxIndex = arr.findMax(nElems);
                              System.out.println("Index of max value: " + maxIndex);
                               arr.removeMax(nElems);
                               nElems--;

                               for ( j = 0; j < nElems; j++)
                                         System.out.print(arr.getElem(j) + " ");
                               System.out.println("");
                    }
          }
```

Lab 2

```
package test;

class Link {
            public int value; // integer data
            public Link next; // reference to next link
          }

          class LinkList {
            private Link first; // ref to first Link on list

            public LinkList() {
              first = null;
            }

            public static Link getnode(int value) {
              Link n = new Link();
              n.value = value;
              n.next = null;
              return n;
            }

            public void insertFirst(int value) {
              Link newLink = getnode(value);
              newLink.next = first; // it points to old first link
              first = newLink; // now first points to this
            }

            public boolean islempty() {
              return (first == null);
            }

            public void printlst() {
              Link ptr = first;
              while (ptr != null) {
                System.out.print(ptr.value + " ");
                ptr = ptr.next; // next node
              }
            }

            // Method to find the max value in the linked list
            public int findMax() {
              if (islempty()) {
                return -1; // Linked list is empty
              }
              Link current = first;
              int max = current.value;
              while (current != null) {
                if (current.value > max) {
                  max = current.value;
                }
                current = current.next;
              }
              return max;
            }

            // Method to remove the max value from the linked list
            public Integer removeMax() {
              if (islempty()) {
                System.out.println("Cannot find max. LinkedList is empty.");
                return null;
              }
              Link current = first;
              Link previous = null;
              Link maxNode = first;
              Link maxPrev = null;

              // Find the max node
              while (current != null) {
                if (current.value > maxNode.value) {
                  maxNode = current;
                  maxPrev = previous;
                }
                previous = current;
                current = current.next;
```

```
    }

    // If maxNode is the first node
    if (maxPrev == null) {
      first = first.next;
    } else {
      maxPrev.next = maxNode.next;
    }

    System.out.println(maxNode.value + " has been removed.");
    return maxNode.value;
  }
}

public class LabLinkedList {
  public static void main(String[] args) {
    LinkList numlist = new LinkList();

    numlist.insertFirst(28);
    numlist.insertFirst(96);
    numlist.insertFirst(75);
    numlist.insertFirst(162);

    numlist.printlst();
    System.out.println();

    System.out.println("Max Value: " + numlist.findMax());

    numlist.removeMax();
    numlist.printlst();
  }
}
```

Lab 3

```
package test;

import java.util.Scanner;

class StackA{
          public int maxSize = 5;
          public String[] stackArray;
          public int top;

          public StackA(int s) {
                    maxSize = s;
                    stackArray = new String[maxSize];
                    top = -1;
          }

          public void push(String j) {
                    if(!isFull()) {
                              stackArray[++top] = j;
                    }else {
                              System.out.println("Stack is full. Cannot push " + j);
                    }
          }
          public String pop() {
                    if(!isEmpty()) {
                              return stackArray[top--];
                    }else {
                              return "Stack is empty";
                    }
          }
          public boolean isEmpty() {
        return (top == -1);
    }

    // Check if stack is full
    public boolean isFull() {
        return (top == maxSize - 1);
    }
}

public class Lab3ForStd {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        StackA stack = new StackA(5); // create stack with max size of 5

        // Accept 5 characters from user and push them onto the stack
        for (int i = 0; i < 5; i++) {
            System.out.print("Push Data " + (i + 1) + ": ");
            String input = sc.nextLine();
            stack.push(input);
        }

        System.out.println("--------------------------");

        // Pop and display the characters in reverse order
        if (!stack.isEmpty()) {
            for (int i = 0; i < 5; i++) {
                System.out.println("Pop Data " + (i + 1) + ": " + stack.pop());
            }
        }

        sc.close();
    }
}
```

Lab 4

```
package test;
import java.util.Scanner;
import java.util.Scanner;

public class Lab4ForStd {
    public static int maxsize = 5;
    public static int[] number = new int[maxsize];
    public static int front = 0;
    public static int rear = -1;
    public static int count = 0;

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        for (int i = 0; i < maxsize; i++) {
            System.out.print("Enqueue Data " + (i + 1) + ": ");
            int num = sc.nextInt();
            enqueue(num);
        }
        enqueue(1000);

        for (int i = 0; i < maxsize; i++) {
            dequeue();
        }
        dequeue(); // This will print "Queue is empty."
    }

    public static void enqueue(int num) {
        if (count < maxsize) {
            rear = (rear + 1) % maxsize;
            number[rear] = num;
            count++;
        } else {
            System.out.println("Queue is full.");
        }
    }

    // Dequeue operation
    public static void dequeue() {
        if (count > 0) {
            System.out.println("Dequeue Data " + count + ": " +
number[front]);
            front = (front + 1) % maxsize;
            count--;
        } else {
            System.out.println("Queue is empty.");
        }
    }
}
```

Lab 5

```
package lab5;

// Lab5 - Bubble Sort
// ข้อ 2: โปรแกรมสำหรับเรียงลำดับข้อมูลแบบ Bubble Sort

import java.util.Scanner; // นำเข้า Scanner เพื่อรับข้อมูลจากผู้ใช้

public class BubbleSort {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in); // สร้าง Scanner สำหรับรับข้อมูลจากผู้ใช้

        // รับจำนวนตัวเลขที่ต้องการเรียงลำดับ
        System.out.print("Input number of integers to sort: ");
        int n = scanner.nextInt(); // อ่านจำนวนตัวเลขที่ผู้ใช้ต้องการเรียงลำดับ
        int[] data = new int[n]; // สร้างอาร์เรย์ขนาด n เพื่อเก็บตัวเลขที่ผู้ใช้ป้อน

        // รับค่าตัวเลขจากผู้ใช้
        System.out.println("Enter " + n + " integers"); // แจ้งให้ผู้ใช้ป้อนตัวเลข
        for (int i = 0; i < n; i++) {
            System.out.print("Enter " + (i + 1) + ": "); // แสดงหมายเลขตัวเลขที่กำลังป้อน
            data[i] = scanner.nextInt(); // อ่านค่าตัวเลขที่ผู้ใช้ป้อนแล้วเก็บไว้ในอาร์เรย์
        }

        // Bubble Sort Algorithm
        // วนลูปสำหรับเรียงลำดับข้อมูล
        for (int i = 0; i < data.length - 1; i++) {
            // ลูปย่อยสำหรับเปรียบเทียบและสลับตำแหน่งตัวเลข
            for (int j = 0; j < data.length - 1 - i; j++) {
                // เปรียบเทียบค่าของตัวเลขสองตำแหน่งติดกัน -i ช่วยลดจำนวนรอบเปรียบเทียบในแต่ละรอบ เพราะค่าที่เรียงถูกต้องจะเลื่อนไปอยู่ตำแหน่งท้ายสุด.
                if (data[j] < data[j + 1]) { // หากตัวเลขตำแหน่งแรก j น้อยกว่าตำแหน่งถัดไป j+1
                    // สลับตำแหน่งตัวเลขเพื่อให้อยู่ในลำดับจากมากไปน้อย
                    int temp = data[j]; // เก็บค่าตำแหน่งแรกในตัวแปรชั่วคราว
                    data[j] = data[j + 1]; // ย้ายค่าจากตำแหน่งถัดไปมาที่ตำแหน่งแรก
                    data[j + 1] = temp; // ย้ายค่าที่เก็บไว้ชั่วคราวไปที่ตำแหน่งถัดไป
                }
            }
        }

        // แสดงผลลัพธ์หลังการเรียงลำดับ
        System.out.println("Sorted list of numbers:"); // แจ้งว่าเป็นผลลัพธ์หลังเรียงลำดับ
        for (int i = 0; i < data.length; i++) {
            // แสดงตัวเลขเรียงลำดับทีละตัวพร้อมระบุหมายเลข
            System.out.println("Data " + (i + 1) + ": " + data[i]);
        }
    }
```

```java
package lab5;

// Lab5 - Selection Sort
// ข้อ 1: โปรแกรมสำหรับเรียงลำดับข้อมูลแบบ Selection Sort

public class SelectionSort {
    public static void main(String[] args) {
        // กำหนดอาร์เรย์ข้อมูลเริ่มต้น
        int[] data = {120, 60, 20, 80, 79, 30, 45}; // ตัวเลขที่ต้องการเรียงลำดับ

        // แสดงข้อมูลก่อนเรียงลำดับ
        System.out.print("Unsorted Array: ["); // แสดงข้อความ "Unsorted Array"
        for (int i = 0; i < data.length; i++) {
            System.out.print(data[i]); // แสดงค่าของอาร์เรย์แต่ละตำแหน่ง
            if (i < data.length - 1) {
                // เช็คว่าตัวเลขปัจจุบันไม่ใช่ตัวสุดท้าย
                // เพื่อเพิ่มเครื่องหมาย ',' คั่นระหว่างตัวเลข
                System.out.print(", ");
            }
        }
        System.out.println("]"); // ปิดวงเล็บ

        // เรียงลำดับข้อมูลด้วย Selection Sort
        for (int i = 0; i < data.length - 1; i++) {
            // ลูปภายนอก: ใช้เลือกตำแหน่งเริ่มต้นของการเปรียบเทียบ
            int minIndex = i; // เก็บตำแหน่งของค่าที่เล็กที่สุดในรอบนี้
            for (int j = i + 1; j < data.length; j++) {
                // ลูปภายใน: ใช้ค้นหาค่าที่เล็กที่สุดในส่วนที่เหลือของอาร์เรย์
                if (data[j] < data[minIndex]) {
                    // เปรียบเทียบค่าที่ตำแหน่ง j กับตำแหน่ง minIndex
                    // หากค่าที่ตำแหน่ง j น้อยกว่าค่าที่ตำแหน่ง minIndex
                    // จะอัปเดตตำแหน่ง minIndex ให้เป็นตำแหน่ง j
                    minIndex = j;
                }
            }
            // สลับตำแหน่งค่าที่เล็กที่สุดกับค่าปัจจุบัน
            int temp = data[minIndex]; // เก็บค่าที่ตำแหน่ง minIndex ไว้ในตัวแปรชั่วคราว
            data[minIndex] = data[i]; // ย้ายค่าปัจจุบันไปที่ตำแหน่ง minIndex
            data[i] = temp; // ย้ายค่าที่เก็บไว้ชั่วคราวไปที่ตำแหน่งปัจจุบัน
        }

        // แสดงข้อมูลหลังเรียงลำดับ
        System.out.print("Sorted Array in Ascending Order: ["); // แสดงข้อความ "Sorted Array in Ascending Order"
        for (int i = 0; i < data.length; i++) {
            System.out.print(data[i]); // แสดงค่าของอาร์เรย์ที่เรียงลำดับแล้ว
            if (i < data.length - 1) {
                // เช็คว่าตัวเลขปัจจุบันไม่ใช่ตัวสุดท้าย
                // เพื่อเพิ่มเครื่องหมาย ',' คั่นระหว่างตัวเลข
                System.out.print(", ");
            }
        }
        System.out.println("]"); // ปิดวงเล็บ
    }
}
```

Lab 6

```
package lab6;

// Lab6 - Binary Search
// ข้อ 2: โปรแกรมสำหรับค้นหาข้อมูลแบบ Binary Search

import java.util.Scanner;

public class BinarySearch {
    public static void main(String[] args) {
        // กำหนดอาร์เรย์ข้อมูล (เรียงลำดับแล้ว)
        int[] Number = {0, 11111, 22222, 33333, 44444, 55555, 66666, 77777, 88888, 99999};

        Scanner scanner = new Scanner(System.in);

        // รับค่าที่ต้องการค้นหา
        System.out.print("Search ID : ");
        int target = scanner.nextInt();

        // Binary Search Algorithm
        int first = 0; // ตำแหน่งเริ่มต้น
        int last = Number.length - 1; // ตำแหน่งสุดท้าย
        boolean found = false; // ตัวแปรสำหรับเช็คว่าพบข้อมูลหรือไม่

        while (first <= last) {
            System.out.println("first: " + first + " last: " + last); // แสดงตำแหน่ง first และ last ในแต่ละรอบ

            int mid = (first + last) / 2; // สูตรคำนวณตำแหน่งตรงกลาง

            if (Number[mid] == target) { // ถ้าค่าที่ตำแหน่ง mid ตรงกับค่าที่ต้องการค้นหา
                System.out.println("Found at Number[" + mid + "] = " + Number[mid]);
                found = true;
                break; // หยุดการค้นหา
            } else if (Number[mid] < target) {
                first = mid + 1; // ปรับตำแหน่ง first ถ้าค่าตรงกลางน้อยกว่าเป้าหมาย
            } else {
                last = mid - 1; // ปรับตำแหน่ง last ถ้าค่าตรงกลางมากกว่าเป้าหมาย
            }
        }

        if (!found) {
            System.out.println("Not Found!");
        }
    }
}
```

```java
package lab6;

// Lab6 - Sequential Search
// ข้อ 1: โปรแกรมสำหรับค้นหาข้อมูลแบบ Sequential Search

import java.util.Scanner;

public class SequentialSearch {
    public static void main(String[] args) {
        // กำหนดอาร์เรย์ข้อมูล
        int[] A = {10001, 10022, 10060, 11255, 15022, 20001, 21002, 23003, 25566, 30078, 40000, 50012, 66006};

        Scanner scanner = new Scanner(System.in);

        // รับค่าที่ต้องการค้นหา
        System.out.print("ID : ");
        int target = scanner.nextInt();

        // Sequential Search Algorithm
        int index = -1; // เก็บตำแหน่งของข้อมูล (ค่าเริ่มต้นคือ -1 เพื่อบอกว่าไม่พบ)
        for (int i = 0; i < A.length; i++) {
            if (A[i] == target) { // ถ้าค่าที่ตำแหน่ง i ตรงกับค่าที่ต้องการค้นหา
                index = i; // เก็บตำแหน่งที่พบ
                break; // หยุดการค้นหา
            }
        }

        // แสดงผลลัพธ์
        if (index != -1) {
            System.out.println("Found at A[" + index + "]");
        } else {
            System.out.println("Not Found!");
        }
    }
}
```

Lab 7

```
package lab7;

import java.util.*;

class BST_class {
    // คลาสย่อย Node สำหรับเก็บข้อมูลแต่ละโหนดของต้นไม้
    class Node {
        int key;
        Node left, right;

        public Node(int data){
            key = data;
            left = right = null;
        }
    }

    Node root;

    // Constructor สำหรับต้นไม้ BST
    BST_class(){
        root = null;  //เป็นโหนดแรกของต้นไม้ (เริ่มต้นเป็น null)
    }

    // ฟังก์ชันสำหรับเพิ่มโหนดใหม่ในต้นไม้
    void insert(int key) {
        root = insert_Recursive(root, key);
    }

    // ฟังก์ชันแบบ recursive สำหรับแทรกโหนดใหม่
    Node insert_Recursive(Node root, int key) {
        if (root == null) {
            root = new Node(key);
            return root;
        }
        if (key < root.key)
            root.left = insert_Recursive(root.left, key);
        else if (key > root.key)
            root.right = insert_Recursive(root.right, key);
        return root;
    }

    // ฟังก์ชัน inorder traversal
    void inorder() {
        System.out.println("In order:");
        inorder_Recursive(root);
        System.out.println(); // เว้นบรรทัด
    }
```

```
    void inorder_Recursive(Node root) {
        if (root != null) {
            inorder_Recursive(root.left); // เข้าถึงโหนดฝั่งซ้าย
            System.out.print(root.key + " "); // แสดงค่าของโหนดปัจจุบัน
            inorder_Recursive(root.right); // เข้าถึงโหนดฝั่งขวา
        }
    }

    // ฟังก์ชัน postorder traversal
    void postorder() {
        System.out.println("Post order:");
        postorder_Recursive(root);
        System.out.println(); // เว้นบรรทัด
    }

    void postorder_Recursive(Node root) {
        if (root != null) {
            postorder_Recursive(root.left); // เข้าถึงโหนดฝั่งซ้าย
            postorder_Recursive(root.right); // เข้าถึงโหนดฝั่งขวา
            System.out.print(root.key + " "); // แสดงค่าของโหนดปัจจุบัน
        }
    }

    // Main method สำหรับรันโปรแกรม
    public static void main(String[] args) {
        BST_class bst = new BST_class();
        Scanner input = new Scanner(System.in); // ใช้รับข้อมูลจากผู้ใช้

        System.out.println("Please enter 10 integers :");

        // รับตัวเลข 10 ตัวจากผู้ใช้และเพิ่มใน BST
        for (int i = 0; i < 10; i++) {
            int num = input.nextInt(); // รับค่าจากผู้ใช้
            bst.insert(num); // เพิ่มตัวเลขในต้นไม้
        }

        // แสดงผลลัพธ์การ traversal แบบ Inorder และ Postorder
        bst.inorder();
        bst.postorder();
    }
}
```

Lab 8

```
package lab8;

import java.util.*;

class Hashing {
    private int[] hash_table = new int[10]; // ตารางแฮชขนาด 10 ช่อง

    // Constructor: ใช้กำหนดค่าเริ่มต้นให้ตารางแฮชทุกช่องว่าง (-1)
    Hashing(){
        for (int j = 0; j < 10; j++) {
            hash_table[j] = -1; // ค่า -1 แสดงว่าช่องนั้นว่าง
        }
    }

    // Method สำหรับแสดงตารางแฮช
    public void display() {
        System.out.print("Table: ");
        for (int i = 0; i < 10; i++) {
            if (hash_table[i] == -1) {
                System.out.print("* "); // ถ้าช่องว่างให้แสดง '*'
            } else {
                System.out.print(hash_table[i] + " "); // ถ้ามีค่าให้แสดงค่านั้น
            }
        }
        System.out.println(); // ขึ้นบรรทัดใหม่
    }

    // Method สำหรับใส่ข้อมูลลงในตารางแฮชโดยใช้ Linear Probing
    public void insert(int key) {
        int index = key % 10; // ใช้แฮชฟังก์ชัน key % 10 เพื่อหาช่องที่จะใส่ข้อมูล
        int originalIndex = index; // เก็บตำแหน่งเริ่มต้นเพื่อป้องกันการวนซ้ำไม่รู้จบ

        while (hash_table[index] != -1) { // ถ้าช่องนั้นมีข้อมูลแล้ว (เกิด Collision หาช่องว่าง)
            index = (index + 1) % 10; // เลื่อนไปที่ช่องถัดไป (แบบวนกลับ)
            if (index == originalIndex) { // ถ้าเลื่อนจนวนกลับมาที่เดิม
                System.out.println("Hash table is full. Cannot insert key: " + key);
                return; // หยุดการทำงาน เพราะไม่มีช่องว่างให้ใส่ข้อมูล
            }
        }

        hash_table[index] = key; // ใส่ข้อมูลลงในช่องที่ว่าง
    }

    // Main method สำหรับรับค่าข้อมูลและเรียกใช้ method อื่น ๆ
    public static void main(String[] args) {
        Hashing hashtab = new Hashing(); // สร้างออบเจกต์สำหรับตารางแฮช
        Scanner input = new Scanner(System.in); // ใช้ Scanner รับค่าจากผู้ใช้

        System.out.println("Please enter 7 integers:");

        // รับค่าตัวเลข 7 ตัวจากผู้ใช้
        for (int i = 0; i < 7; i++) {
            int num = input.nextInt(); // รับค่าตัวเลข
            hashtab.insert(num); // ใส่ตัวเลขลงในตารางแฮช
        }

        // แสดงผลตารางแฮชหลังจากใส่ข้อมูลเสร็จ
        hashtab.display();
    }
}
```

Lab 9

```
package graph;

public class GraphForSTD {

public static void main(String args[]) {

        // Object of graph is created.
        Graphs<Integer> g = new Graphs<Integer>();

        // (1) Setup graph details
        // ส่วนนี้เพิ่มเส้นเชื่อม (Edge) ระหว่าง Vertex ต่างๆ ในกราฟ
        g.addEdge(0, 1, true); // เพิ่มเส้นเชื่อมระหว่าง Vertex 0 และ 1 แบบสองทาง
        g.addEdge(0, 4, true);
        g.addEdge(1, 2, true);
        g.addEdge(1, 3, true);
        g.addEdge(1, 4, true);
        g.addEdge(2, 3, true);
        g.addEdge(3, 4, true);

        // Print adjacency list
        System.out.println("Graph:");
        System.out.println(g.toString());

        // (2.1) Display the number of vertices
        g.getVertexCount();

        // (2.2) Display the number of edges
        // true หมายถึงเป็นกราฟแบบสองทาง
        g.getEdgesCount(true);

        // (2.3) Check if an edge exists between vertex 3 and 4
        g.hasEdge(3, 4);

        // (2.4) Check if vertex 5 exists
        g.hasVertex(5);
    }
}
```

```java
package graph;

import java.util.*;

class Graphs<T> {

    // ใช้HashMap ในการเก็บโครงสร้างข้อมูลของกราฟ โดย Key เป็น Vertex และ Value เป็น List ของ Vertex ที่เชื่อมต่อ
    private Map<T, List<T>> map = new HashMap<>();

    // ฟังก์ชันนี้ใช้เพิ่ม Vertex ใหม่ในกราฟ
    public void addVertex(T s) {
        map.put(s, new LinkedList<T>()); // เพิ่ม Vertex ใหม่พร้อมรายการว่างของ Vertex ที่เชื่อมต่อ
    }

    // This function adds the edge between source to destination
    // ฟังก์ชันนี้ใช้เพิ่มเส้นเชื่อมระหว่าง Source และ Destination
    public void addEdge(T source, T destination, boolean bidirectional) {
        if (!map.containsKey(source)) // หากยังไม่มี Source ในกราฟ ให้เพิ่ม
            addVertex(source);
        if (!map.containsKey(destination)) // หากยังไม่มี Destination ในกราฟ ให้เพิ่ม
            addVertex(destination);
        map.get(source).add(destination); // เพิ่มเส้นเชื่อมจาก Source ไป Destination
        if (bidirectional) {
            map.get(destination).add(source); // หากเป็นกราฟแบบสองทาง ให้เพิ่มเส้นเชื่อมกลับกัน
        }
    }

    // (2.1) Function to display the number of vertices
    // ฟังก์ชันนี้ใช้แสดงจำนวน Vertex ทั้งหมดในกราฟ
    public void getVertexCount() {
        int vertexCount = map.size(); // นับจำนวน Key ใน HashMap
        if (vertexCount == 1) {
            System.out.println("The graph has 1 vertex.");
        } else {
            System.out.println("The graph has " + vertexCount + " vertices.");
        }
    }

    // (2.2) Function to display the number of edges
    // ฟังก์ชันนี้ใช้แสดงจำนวนเส้นเชื่อม (Edge) ทั้งหมดในกราฟ
    public void getEdgesCount(boolean bidirectional) {
        int count = 0;
        for (T v : map.keySet()) {
            count += map.get(v).size(); // นับจำนวน Vertex ที่เชื่อมต่อจากแต่ละ Vertex
        }
        if (bidirectional) {
            count /= 2; // หาร 2 หากเป็นกราฟแบบสองทาง
        }
        if (count == 1) {
            System.out.println("The graph has 1 edge.");
        } else {
            System.out.println("The graph has " + count + " edges.");
        }
    }

    // (2.4) Function to check if a vertex exists
    // ฟังก์ชันนี้ใช้ตรวจสอบว่า Vertex ที่กำหนดมีอยู่ในกราฟหรือไม่
    public void hasVertex(T s) {
        if (map.containsKey(s)) {
            System.out.println("The graph contains " + s + " as a vertex.");
        } else {
            System.out.println("The graph does not contain " + s + " as a vertex.");
        }
    }

    // (2.3) Function to check if an edge exists between two vertices
    // ฟังก์ชันนี้ใช้ตรวจสอบว่ามีเส้นเชื่อมระหว่าง Vertex สองตัวหรือไม่
    public void hasEdge(T s, T d) {
        if (map.get(s).contains(d)) {
            System.out.println("The graph has an edge between vertex " + s + " and " + d + ".");
        } else {
            System.out.println("The graph has no edge between " + s + " and " + d + ".");
        }
    }

    // ฟังก์ชันนี้แสดง Adjacency List ของกราฟ
    @Override
    public String toString() {
        StringBuilder builder = new StringBuilder();
        for (T v : map.keySet()) {
            builder.append(v.toString() + ": "); // เพิ่ม Vertex หลัก
            for (T w : map.get(v)) {
                builder.append(w.toString() + " "); // เพิ่ม Vertex ที่เชื่อมต่อ
            }
            builder.append("\n");
        }
        return builder.toString();
    }
}
```