| **Data Structures and Algorithms Laboratory** | | |
|---|---|---|
| **Laboratory 4:** Circular Linked Lists and Algorithm Analysis | **School of Information Technology** | |
| **Name:** Pattarapon Bunchuai | **ID:**6431503044 | **Section:**3 |
| **Date:** | **Due date:** **on LMS** | |

**Objective**

- To analyze algorithms based on experimental methods
- To implement circular linked lists

**Exercise 1:** (In-class) Use experimental analysis to compare the following two algorithms. Fill in the times from your experiments and plot the graphs of data size (n) versus time (ms).

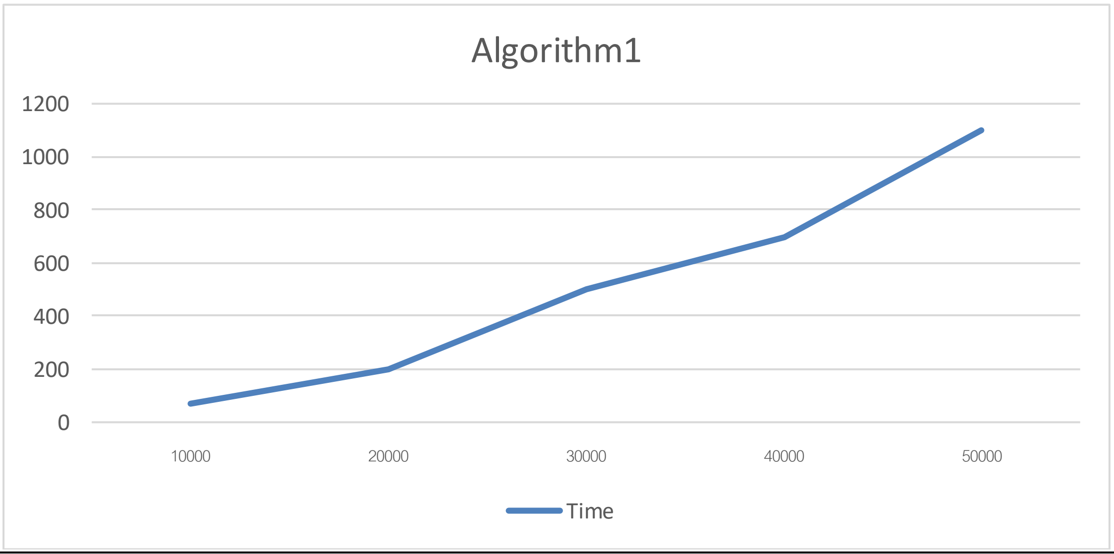
**Algorithm 1**

Assume that the data size (n) is from 10000 to 50000

```
for(int i=1;i<=n;i++) {
      for(int j=1;j<=n;j++) {
       result = i+j;
      }
}
```

| n | 10000 | 20000 | 30000 | 40000 | 50000 |
|---|---|---|---|---|---|
| Time (ms) | 70 | 200 | 500 | 700 | 1100 |

Capture a graph image and paste here.



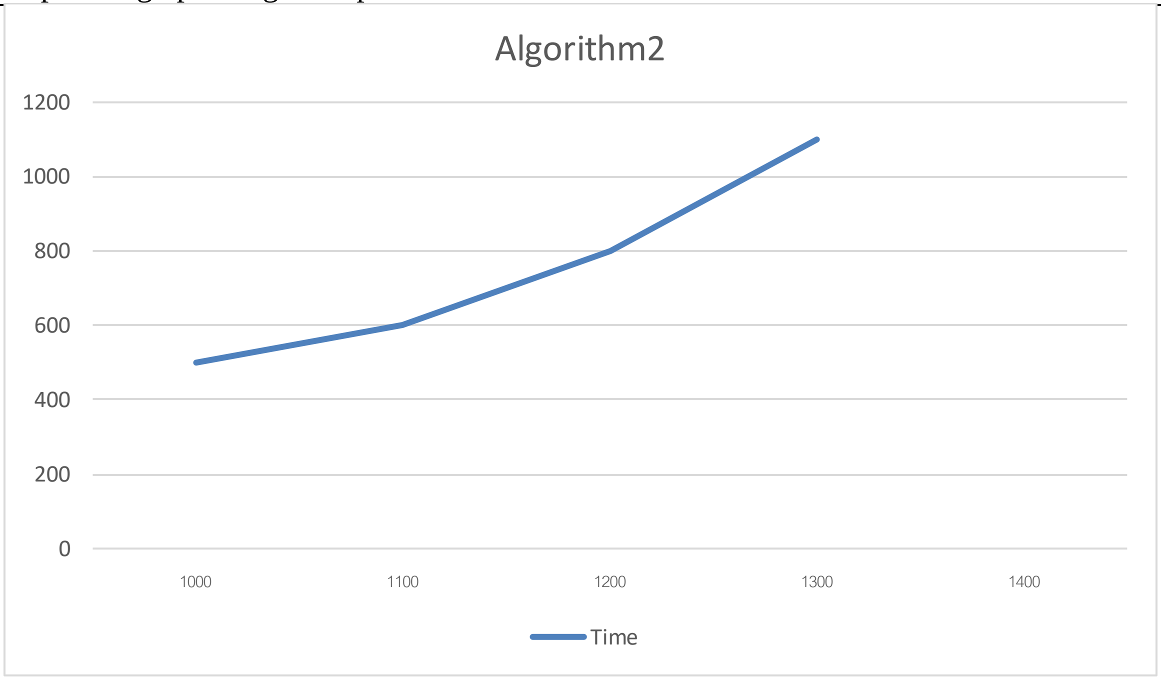
**Algorithm 2**

Assume that the data size (n) is from 1000 to 1400

```
for(int i=1;i<=n;i++) {
      for(int j=1;j<=n;j++) {
            for(int k=1;k<=n;k++) {
             result = i+j+k;
            }
      }
}
```

| n | 1000 | 1100 | 1200 | 1300 | 1400 |
|---|---|---|---|---|---|
| Time (ms) | 5000 | 600 | 800 | 1100 | 1300 |

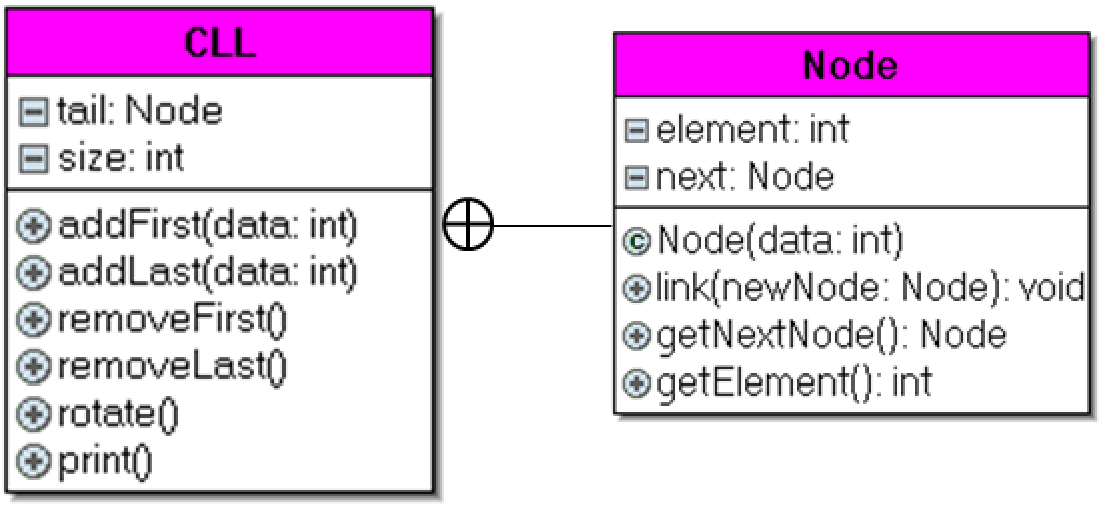
Capture a graph image and paste here.



**Which algorithm is faster? Explain.**

Algorithm1 is faster than algorithm2 because look at the number of n and the time when execute the program algorithm1 take less time to execute the program at high number than algorithm1.

**Exercise 2:** (Homework) From the given class diagram, create a circular linked list and complete the program to get the results as shown.



<u>Expected result</u>

```
Empty linked list
->1->
->1->2->
->3->1->2->
->1->2->3->
->2->3->
->2->
```

```
package lab.home_work;

//================= CLL class ========================
class CLL {
        // ---------------------- Node -------------------
        private class Node {
                private int element;
                private Node next;

                // constructor
                public Node(int data) {
                        element = data;
                        next = null;
                }

                // link a new node to this node
                public void link(Node newNode) {
                        next = newNode;
                }
                //return next node
                public Node getNextNode() {
                        return next;
                }

                //return element of this node
                public int getElement() {
                        return element;
                }
        }
        //-------------------- End Node --------------------

        //CLL properties and methods
        private Node tail = null;
        private int size = 0; // SLL's size

        public void addFirst(int data) {
                Node newNode = new Node(data);

                if (size == 0) {
                        tail = newNode;
                } else {
                        newNode.link(tail.getNextNode());
                }
                tail.link(newNode);
                size++;
        }

        public void addLast(int data) {
                Node newNode = new Node(data);

                if (size == 0) {
                        tail = newNode;
                } else {
                        newNode.link(tail.getNextNode());
                }
                tail.link(newNode);
                tail = newNode;
                size++;
        }

        public void removeFirst() {
                if (size == 0) {
                        return;
                }

                if (size == 1) {
                        tail = null;
                        size = 0;
                        return;
                }

                tail.link(tail.getNextNode().getNextNode());
                size--;
        }

        public void removeLast() {
                if (size == 0) {
                        return;
                }

                if (size == 1) {
                        tail = null;
                        size = 0;
                        return;
                }

                //If have one more node
                for (Node p = tail.getNextNode(); p != tail; p = p.getNextNode()) {
                        if (p.getNextNode() == tail) {
                                p.link(tail.getNextNode());
                                tail = p;
                                size --;
                                break;
                        }
                }

        }

        public void rotate() {
                if (size <= 1) {
                        return;
                }

                tail = tail.getNextNode();
        }

        public void print() {
                if (size == 0) {
                        System.out.println("Empty linked list");
                } else {
                        System.out.print("->");
                        for (Node p = tail.getNextNode(); p != tail; p = p.getNextNode()) {
                                System.out.print(p.getElement() + "->");
                        }
                        System.out.println(tail.getElement() + "->");
                }
        }
}
```

```
//================== MainCLL class =========================
public class MainCLL {
	public static void main(String[] args) {
		CLL cll = new CLL();
		cll.print();
		cll.addFirst(1);
		cll.print();
		cll.addLast(2);
		cll.print();
		cll.addFirst(3);
		cll.print();
		cll.rotate();
		cll.print();
		cll.removeFirst();
		cll.print();
		cll.removeLast();
		cll.print();
	}
}
```