1. (**실습**: 이진탐색트리) 다음은 이진탐색트리를 생성하는 코드이다. 아래 코드를 입력하고 실행하면서 이진탐색트리 노드 삽입법을 익히시오. (Reference: https://algs4.cs.princeton.edu/32bst/BST.java.html, GPLv3)

   - **실습** #1: 아래 search 함수는 비재귀적으로 이진탐색을 수행한다. 이진탐색을 재귀적으로 수행하도록 다음 형식의 재귀호출함수 searchRecur()를 구현하시오.
   public BinaryTree search(int key) { return searchRecur(root, key); }
   private BinaryTree searchRecur(BinaryTree node, int key) { … }

   - **실습** #2: 아래 add(int key) 함수는 재귀호출함수 add(BinaryTree tree, int key)를 통해 재귀적으로 구현되었다. add(int key) 함수를 비재귀적 방식으로 구현하시오.

   - **실습** #3: 아래 main 함수의 마지막 문장이 정상 동작하도록 BinarySearchTree 클래스 내 inorder 함수를 구현하시오. inorder 함수는 이진탐색트리의 노드들을 inorder 순으로 방문하여 모든 key 값들을 문자열로 반환하는 함수이다. 아래 main 함수 마지막 문장의 실행 결과는 다음과 같다.
   5 10 15 20 25 30 35 50 60 62 64 65 70 90

```
public class Test {
	public static void main(String[] args) {
		BinarySearchTree	tree=new BinarySearchTree();
		int		n[]={50,20,70,10,30,5,15,25,60,90,62,65,64,35};
		for (int i = 0; i < n.length; i++) tree.add(n[i]);
		System.out.println(tree.search(30));
		System.out.println(tree.search(33));
		System.out.println(tree);
		//System.out.println(BinarySearchTree.inorder(tree.root));
	}
}
class BinarySearchTree {
	class BinaryTree {
		int		key;
		BinaryTree		left, right;
		public BinaryTree(int key) { this.key=key; }
		@Override
		public String toString() { return Integer.toString(key); }
	}
	BinaryTree		root;
	public void add(int key) {
		root=add(root, key);
	}
	private BinaryTree add(BinaryTree		tree, int key) {
		if(tree==null) return new BinaryTree(key);
		if(tree.key<key) tree.right=add(tree.right, key);
		else if(tree.key>key) tree.left=add(tree.left, key);
		else ; // value 삽입 시 else tree.value=value;
		return tree;
	}
	public BinaryTree search(int key) {
		BinaryTree		node=root;
		while(node!=null){
			if(node.key==key) return node;
			if(node.key<key) node=node.right;
			else node=node.left;
		}
		return node;
	}
	@Override
	public String toString() {
		return levelOrder().toString();
	}
	private LinkedList<BinaryTree> levelOrder() {
		LinkedList<BinaryTree>		list=new LinkedList<>();
		LinkedList<BinaryTree>		queue=new LinkedList<>();
		if(root!=null) queue.addLast(root);
		while(!queue.isEmpty()){
			BinaryTree		node=queue.removeFirst();
			list.add(node);
			if(node.left!=null) queue.addLast(node.left);
			if(node.right!=null) queue.addLast(node.right);
		}
		return list;
	}
}
```

2. (자바클래스 이진탐색트리) 다음은 자바클래스 TreeSet를 활용한 예시 코드이다. TreeSet은 red-black tree에 기반한 균형이진탐색트리를 구현한 자바클래스이다. 아래 코드를 입력하고 실행하면서 자바클래스 TreeSet의 사용법을 학습하시오.

```
public class Test {
	public static void main(String[] args) {
		int		n[]={50,20,70,10,30,5,15,25,60,90,62,65,64,35};
		TreeSet<Integer>	tree=new TreeSet<>();
		for (int i = 0; i < n.length; i++) tree.add(n[i]);
		System.out.println(tree.contains(30));
		System.out.println(tree.contains(33));
		System.out.println(tree);
	}
}
```

3. (자바클래스 이진탐색트리) 다음은 자바클래스 TreeMap을 활용한 예시 코드이다. TreeMap은 red-black tree에 기반한 균형이진탐색트리를 구현한 자바클래스이다. 아래 코드를 입력하고 실행하면서 자바클래스 TreeMap의 사용법을 학습하시오.

```
public class Test {
	public static void main(String[] args) {
		TreeMap<String, Integer>		map=new TreeMap<>();
		map.put("Korea", 32);
		map.put("Japan", 50);
		map.put("China", 16);
		System.out.println(map);
		map.remove("Japan");
		System.out.println(map);
		System.out.println(map.size());
		System.out.println(map.get("Korea"));
		System.out.println(map.get("Germany"));
	}
}
```

4. (**실습**: 자바클래스 이진탐색트리) 총 백만명의 Player 객체를 TreeMap에 저장(id값을 key로, Player 객체를 value로 저장)한 후, id 값이 홀수인 Player들을 삭제하는 코드를 작성하려고 한다. 백만명의 Player 객체는 각각 1부터 1000000까지의 id 값을 갖는다고 한다. 이 코드를 완성하시오.

```
public class Player {
	int		id;
	public Player(int id) {
			this.id=id;
	}
}
public class Test {
	public static void main(String[] args) {
	}
}
```

References

- C로 쓴 자료구조론 (Fundamentals of Data Structures in C, Horowitz et al.). 이석호 역. 사이텍 미디어. 1993.
- 쉽게 배우는 알고리즘: 관계 중심의 사고법. 문병로. 한빛아카데미. 2013.
- C언어로 쉽게 풀어 쓴 자료구조. 천인국 외 2인. 생능출판사. 2017.
- 김윤명. (2008). 뇌를 자극하는 Java 프로그래밍. 한빛미디어.
- 남궁성. 자바의 정석. 도우출판.
- 김윤명. (2010). 뇌를 자극하는 JSP & Servlet. 한빛미디어.