1. (**실습**: 거품정렬, bubble sort) 다음은 정수 배열을 거품정렬하는 코드 예이다. 아래 코드를 입력하고 실행하면서 거품정렬 구현법을 익히시오. (Reference: https://en.wikipedia.org/wiki/Bubble_sort, CC-BY-SA)

   - **실습** #1: 아래 bubbleSort 함수의 시간 복잡도를 개선하시오. 힌트: 외부 for 루프는 각 반복마다 정렬되지 않은 자료 한 개의 정렬된 위치(입력 배열의 오른쪽 끝부터 시작하여 1씩 감소된 위치)를 결정하므로, 이전 반복을 통해 기정렬된 자료들은 이후 반복 시 재검사할 필요가 없다.
   - **실습** #2: 아래 bubbleSort 함수의 시간 복잡도를 개선하시오. 힌트: 외부 for 루프의 특정 반복 시 교환이 전혀 발생하지 않았다면 자료는 이미 정렬된 상태이므로 더 이상의 반복은 필요 없다.

```
public class Test {
	public static void main(String[] args) {
		int	v[]=new int[]{5,8,1,9,3,5,1,5};
		bubbleSort(v);
		System.out.println(Arrays.toString(v));
	}
	private static void bubbleSort(int[] v) {
		for (int i = 0; i < v.length; i++) {
			for (int j = 1; j < v.length; j++) {
				if(v[j-1]>v[j]) { // 인접 자료 간 순서 맞지 않으면 교환
					int	temp=v[j];
					v[j]=v[j-1];
					v[j-1]=temp;
				}
			}
		}
	}
}
```

2. (**실습**: 선택정렬, selection sort) 다음은 정수 배열을 선택정렬하는 코드 예이다. 아래 코드를 입력하고 실행하면서 선택정렬 구현법을 익히시오. (Reference: https://en.wikipedia.org/wiki/Selection_sort, CC-BY-SA)

- 선택정렬은 정렬되지 않은 리스트 L 내 최소값 위치를 찾아 L의 첫 위치 자료와 교환한 후 L의 첫 위치를 오른쪽으로 하나 이동하는 작업을 반복함으로써 정렬을 수행한다.
- **실습** #1: 선택정렬에서 첫 위치 자료가 최소값인 경우에는 교환이 불필요하다. 아래 selectionSort 함수에 이를 반영하시오.

```
public class Test {
	public static void main(String[] args) {
		int	v[]=new int[]{5,8,1,9,3,5,1,5};
		selectionSort(v);
		System.out.println(Arrays.toString(v));
	}
	private static void selectionSort(int[] v) {
		for (int i = 0; i < v.length-1; i++) {
			int	minPos=i;
			for (int j = i+1; j < v.length; j++) {
				if(v[minPos]>v[j]) minPos=j;
			}
			int	x=v[i]; v[i]=v[minPos]; v[minPos]=x;
		}
	}
}
```

3. (**실습**: 삽입정렬, insertion sort) 다음은 정수 배열을 삽입정렬하는 코드 예이다. 아래 코드를 입력하고 실행하면서 삽입정렬 구현법을 익히시오. (Reference: https://en.wikipedia.org/wiki/Insertion_sort, CC-BY-SA)

- 삽입정렬은 정렬되지 않은 리스트에서 추출된 자료를 정렬된 리스트 내 올바른 위치에 삽입하는 과정을 반복함으로써 정렬을 수행한다.
- 아래 코드에서는 정렬되지 않은 각 자료의 삽입 위치를 찾기 위해 정렬된 자료의 끝 위치 자료부터 크기를 비교하면서 교환하는 작업을 반복한다.
- **실습** #1: 아래 insertionSort의 내부 for 루프의 swap은 세 번의 대입이 발생한다. 이를 개선해 보시오. 힌트: 내부 for 루프 진입 전에 삽입할 자료를 보관해 두고 정렬된 자료의 끝 위치부터 앞으로 이동하면서 삽입할 자료보다 큰 자료인 경우 오른쪽으로 한 칸씩 이동하는 작업을 반복한 후, 반복 종료 지점(삽입 위치)에 보관해 둔 자료를 저장한다.

```
public class Test {
	public static void main(String[] args) {
		int	v[]=new int[]{5,8,1,9,3,5,1,5};
		insertionSort(v);
		System.out.println(Arrays.toString(v));
	}
	private static void insertionSort(int[] v) {
		for (int i = 1; i < v.length; i++) {
			for (int j = i; j>0 && v[j-1]>v[j]; j--) {
				int	x=v[j]; v[j]=v[j-1]; v[j-1]=x;
			}
		}
	}
}
```

4. (**실습**: 정렬된 배열 합병) 다음은 두 정렬된 배열을 하나의 정렬된 배열로 합병하는 코드이다. 이 코드를 완성하시오.

- 구현방법: 두 배열 전체를 담을 수 있는 새로운 배열 v를 만든다. n[0], m[0]를 비교한다. n[0]<m[0]인 경우 n[0]을 v[0]에 대입한다. n[1], m[0]을 비교한다. n[1]>m[0]인 경우 m[0]을 v[1]에 대입한다. n[1], m[1]을 비교한다. n[1]>m[1]인 경우 m[1]을 v[2]에 대입한다. n[1], m[2]를 비교한다. ...
- 실행결과: [1, 2, 3, 4, 5, 6, 7, 7, 8, 10]

```
public class Test {
	public static void main(String[] args) {
		int	n[]={1,3,5,7,8};
		int	m[]={2,4,6,7,10};
		int	v[]=mergeArray(n, m);
		System.out.println(Arrays.toString(v));
	}
	private static int[] mergeArray(int[] n, int[] m) {
	}
}
```

5. (**실습**: 정렬된 배열 합병) 다음은 하나의 배열 내 정렬된 부분 배열들을 합병하여 전체 정렬된 배열로 변환하는 코드이다. 아래 코드에서 v의 두 부분 배열 v[0..4], v[5..9]은 [1,3,5,7,8], [2,4,6,7,10]으로 각각 정렬된 상태이나 v 전체는 정렬되어 있지 않다. 이 코드를 완성하시오.

- 실행결과: [1, 2, 3, 4, 5, 6, 7, 7, 8, 10]

```
public class Test {
	public static void main(String[] args) {
		int	v[]={1,3,5,7,8,2,4,6,7,10};
		mergeArray(v, 0, 4, 5, 9); // v의 부분 배열들 v[0..4], v[5..9]이 정렬된 상태임
		System.out.println(Arrays.toString(v));
	}
	private static void mergeArray(int[] v, int left1, int right1, int left2, int right2) {
	}
}
```

6. (**실습**: 합병정렬, merge sort) 다음은 정수 배열을 합병정렬하는 코드 예이다. 아래 코드를 입력하고 실행하면서 합병정렬 구현법을 익히시오. (Reference: https://en.wikipedia.org/wiki/Merge_sort, CC-BY-SA)

- **실습** #1: 아래 mergeArray 함수를 완성하시오.

```
public class Test {
	public static void main(String[] args) {
		int		v[]=new int[]{8,5,9,1,5,3,5,1};
		mergeSort(v, 0, v.length-1);
		System.out.println(Arrays.toString(v));
	}

	private static void mergeSort(int[] v, int left, int right) {
		if(left==right) return;
		int		m=(left+right)/2;
		mergeSort(v, left, m);
		mergeSort(v, m+1, right);
		mergeArray(v, left, m, m+1, right);
	}
	private static void mergeArray(int[] v, int left1, int right1, int left2, int right2) {
	}
}
```

7. (**실습**: 퀵정렬, quick sort, Lomuto 분할법 기반) 다음은 정수 배열을 퀵정렬하는 코드 예이다. 아래 코드를 입력하고 실행하면서 퀵정렬 구현법을 익히시오. (Reference: https://en.wikipedia.org/wiki/Quicksort, CC-BY-SA)

   - 실습: partition 함수 내 swap 문의 경우 p와 k가 같은 경우 교환 불필요함. 개선 필요

```
public class Test {
	public static void main(String[] args) {
		int	v[]={8,5,9,1,5,3,5,1};
		quickSort(v, 0, v.length-1);
		System.out.println(Arrays.toString(v));
	}
	private static void quickSort(int[] v, int low, int high) {
		if(low>=high) return;
		int	p=partition(v, low, high);
		quickSort(v, low, p-1);
		quickSort(v, p+1, high);
	}
	private static int partition(int[] v, int low, int high) {
		int	pivot=v[high];
		int	p=low; // GTEQ 리스트 첫 위치 (반복 종료 후 pivot 위치임)
		for (int k = low; k < high; k++) {
			if(v[k]<pivot) swap(v, p++, k); // GTEQ 불만족 자료 LT로 이동, GTEQ첫위치 수정
		}
		swap(v, p, high); // GTEQ 리스트 첫 자료와 pivot 교환
		return p;
	}
	private static void swap(int[] v, int k, int i) {
		int	temp=v[k];
		v[k]=v[i];
		v[i]=temp;
	}
}
```

8. (퀵정렬, quick sort, Hoare 분할법 기반) 다음은 정수 배열을 퀵정렬하는 코드 예이다. 아래 코드를 입력하고 실행하면서 퀵정렬 구현법을 익히시오. (Reference: https://en.wikipedia.org/wiki/Quicksort, CC-BY-SA)

```
public class Test {
	public static void main(String[] args) {
		int	v[]={8,5,9,1,5,3,5,1};
		quickSort(v, 0, v.length-1);
		System.out.println(Arrays.toString(v));
	}
	private static void quickSort(int[] v, int low, int high) {
		if(low>=high) return;
		int	p=partition(v, low, high);
		// Lomuto 방법에서는 v[p]가 pivot이 되도록 동작함
		// Hoare 방법에서는 v[p]가 pivot이라는 보장 없음. 즉 quickSort(v,low,p-1)은 불가
		//  반례) 8 2 5 => partition 후 => 5 2 8 (p=1) => [5 2], [8]의 분할임. [5],2,[8]로의 분할 불가
		quickSort(v, low, p);
		quickSort(v, p+1, high);
	}
	private static int partition(int[] v, int low, int high) {
		int	i=low-1, j=high+1, pivot=v[low];
		while(true){
			while(v[++i]<pivot); // stop if pivot<=v[i] (GTEQ)
			while(v[--j]>pivot); // stop if v[j]<=pivot (LTEQ)
			if(i<j) swap(v, i, j);
			else return j; // j<=i    (j: right end of LTEQ list, i: left end of GTEQ list)
		}
	}
	private static void swap(int[] v, int k, int i) {
		int	temp=v[k];
		v[k]=v[i];
		v[i]=temp;
	}
}
```

9. (**실습**) 다음은 배열 v를 문자열 길이의 오름차순으로 정렬한 후 출력하는 코드이다. 아래 bubbleSort 함수를 완성하시오.

- 실행결과 예시: [UK, Korea, China, Canada, Brazil, United States]

```
public class Test {
	public static void main(String[] args) {
			String	v[]={"Korea", "UK", "China", "United States", "Canada", "Brazil"};
			bubbleSort(v);
			System.out.println(Arrays.toString(v));
	}
	private static void bubbleSort(String v[]) {
	}
}
```

10. (**실습**) 다음은 Player 배열 v를 Player record의 오름차순으로 정렬한 후 출력하는 코드이다. 아래 bubbleSort 함수를 완성하시오.

- 실행결과 예시: [(yhkim,4.3), (cslee,7.3), (cskim,9.1), (yhlee,9.8)]

```
public class Player {
	String	id;
	double	record;
	public Player(String id, double record) {
			this.id=id;
			this.record=record;
	}
	@Override
	public String toString() {
			return "("+id+","+record+")";
	}
}
public class Test {
	public static void main(String[] args) {
	    Player  v[]={new Player("yhkim", 4.3),new Player("cskim", 9.1),new Player("yhlee", 9.8),new Player("cslee", 7.3)};
	    bubbleSort(v);
	    System.out.println(Arrays.toString(v));
	}
	private static void bubbleSort(Player v[]) {
	}
}
```

11. 다음은 자바에서 제공하는 Arrays.sort() 함수를 이용하여 배열 v를 문자열 길이의 오름차순으로 정렬한 후 출력하는 코드이다. Comparator 인터페이스의 compare 함수는 LT, EQ, GT에 대해 각각 -1(음의 정수), 0, 1(양의 정수)을 반환하도록 정의된다. 다음 코드를 입력하고 실행하면서 Arrays.sort() 함수를 사용하는 방법을 익히시오.

```
public class Test {
	public static void main(String[] args) {
		String	v[]={"Korea", "UK", "China", "United States", "Canada", "Brazil"};
		Arrays.sort(v, new Comparator<String>() {
			@Override
			public int compare(String o1, String o2) {
				return o1.length()-o2.length();
			}
		});
		System.out.println(Arrays.toString(v));
	}
}
```

12. (**실습**) 다음은 자바에서 제공하는 Arrays.sort() 함수를 이용하여 Player 배열 v를 Player record의 오름차순으로 정렬한 후 출력하는 코드이다. 아래 Arrays.sort() 문장을 완성하시오.

```
public class Player {
	String	id;
	double	record;
	public Player(String id, double record) {
		this.id=id;
		this.record=record;
	}
	@Override
	public String toString() {
		return "("+id+","+record+")";
	}
}
public class Test {
	public static void main(String[] args) {
	    Player  v[]={new Player("yhkim", 4.3),new Player("cskim", 9.1),new Player("yhlee", 9.8),new Player("cslee", 7.3)};
	    Arrays.sort(                    );
	    System.out.println(Arrays.toString(v));
	}
}
```

13. (**실습**: 정수 정렬, 최소/최대값 인지, 비중복데이터) 정수 배열 v에 중복되지 않은 0~100 범위의 임의 값들이 n개 저장되어 있다. 아래 코드는 함수 sort()를 호출하여 배열 v를 정렬한 후 출력한다고 한다. 비교 기반 정렬을 사용하지 않으면서 배열 v을 정렬하는 함수 sort()를 완성하시오.

- 구현 방법: 함수 sort 내에서 크기 101의 새로운 배열 count를 만들고 0으로 초기화한 후 배열 v 내 각 값의 출현 여부를 배열 count에 저장한 다음, 배열 count를 첫 위치부터 검사하면서 출현한 값을 배열 v에 순차 저장한다.

```
public class Test {
	public static void main(String[] args) {
		int	v[]={87,95,53,77,100,14};
		sort(v);
		System.out.println(Arrays.toString(v));
	}
	private static void sort(int[] v) {
	}
}
```

14. (**실습**: 정수 정렬, 최소/최대값 인지, 중복데이터) 정수 배열 v에 중복 허용 0~100 범위의 임의 값들이 n개 저장되어 있다. 아래 코드는 함수 sort()를 호출하여 배열 v를 정렬한 후 출력한다고 한다. 비교 기반 정렬을 사용하지 않으면서 배열 v을 정렬하는 함수 sort()를 완성하시오.

- 구현 방법: 함수 sort 내에서 크기 101의 새로운 배열 count를 만들고 0으로 초기화한 후 배열 v 내 각 값의 출현 횟수를 배열 count에 저장한 다음, 배열 count를 첫 위치부터 검사하면서 출현한 값을 출현 횟수만큼 배열 v에 순차 저장한다.

```
public class Test {
	public static void main(String[] args) {
		int	v[]={87,95,53,77,100,95,14};
		sort(v);
		System.out.println(Arrays.toString(v));
	}
	private static void sort(int[] v) {
	}
}
```

15. (**실습**: 정수 정렬, 중복데이터, 범위 크기 인지) 정수 배열 v에 중복 허용 임의 값(음수 포함)들이 n개 저장되어 있으며 배열 내 저장된 값들의 범위 크기(최대값과 최소값의 차이 + 1)는 100,000을 초과하지 않는다고 한다(아래 예시 참조). 아래 코드는 함수 sort()를 호출하여 배열 v를 정렬한 후 출력한다고 한다. 비교 기반 정렬을 사용하지 않으면서 배열 v을 정렬하는 함수 sort()를 완성하시오. 아래 예시 각 배열 v에 대해 동작하도록 구현하시오.

- 구현 방법: 새로운 배열 count를 만들고 0으로 초기화 한 후 배열 v 내 각 값의 출현 횟수를 배열 count에 저장한다. 배열 count의 크기는 배열 v 내 최소값, 최대값에 기반하여 결정한다. count[0]에는 배열 v의 최소값의 출현 횟수를 저장하고, count[count.length-1]에는 배열 v의 최대값의 출현 횟수를 저장한다.

- 예) int v[]={5,2,2,1,3};
- 예) int v[]={10005,10002,10002,10001,10003};
- 예) int v[]={-10005,-10002,-10002,-10001,-10003};
- 예) int v[]={1,-5,3,2,1,4,-3,-1,-1,0};

```
public class Test {
	public static void main(String[] args) {
		int	v[]={10005,10002,10002,10001,10003};
		sort(v);
		System.out.println(Arrays.toString(v));
	}
	private static void sort(int[] v) {
	}
}
```

16. (계수정렬, Counting sort, max known, 0 or postive integer) 다음은 최대값이 알려진 0이상 양의 정수(int)들을 계수정렬하는 코드의 예이다. 아래 코드를 입력하고 실행하면서 계수정렬 구현법을 익히시오. (Reference: https://rosettacode.org/wiki/Sorting_algorithms/Counting_sort, Licence: GNU FDL, CC-BY-2.5-CA, https://opendatastructures.org/ods-cpp/11_2_Counting_Sort_Radix_So.html CC BY 2.5 CA).

```
public class Test {
	public static void main(String[] args) {
		int	n[]={5,8,1,9,3,5,1,5};
		countingSort(n, 10); // 최대값 10
		System.out.println(Arrays.toString(n));
	}
	private static void countingSort(int[] n, int max) {
		int	m[]=new int[n.length], count[]=new int[max+1];
		for (int i = 0; i < n.length; i++) count[n[i]]++;
		for (int i = 1; i < count.length; i++) count[i]+=count[i-1];
		for (int i = n.length-1; i >=0; i--) m[--count[n[i]]]=n[i];
		for (int i = 0; i < m.length; i++) n[i]=m[i];
	}
}
```

17. (**실습**) 다음은 배열 v를 문자열 길이의 오름차순으로 정렬한 후 출력하는 코드이다. 문자열 최대 길이를 20으로 가정하고 아래 countingSort 함수를 완성하시오.

- 실행결과 예시: [UK, Korea, China, Canada, Brazil, United States]

```
public class Test {
	public static void main(String[] args) {
		String	v[]={"Korea", "UK", "China", "United States", "Canada", "Brazil"};
		countingSort(v, 20); // 문자열 최대 길이를 20으로 가정
		System.out.println(Arrays.toString(v));
	}
	private static void countingSort(String[] n, int max) {
	}
}
```

18. (**실습**: 십진수의 특정 위치 자리 값 추출) 다음은 임의 정수의 특정 위치 자리수 값을 출력하는 코드이다. 아래 코드를 완성하시오. pos=3은 십진수에서 천의 자리수를 의미한다.(pos=0은 1의 자리수).

- 실행 예:

```
1403997200 => 7
1346619619 => 9
1829013463 => 3
```

```
public class Test {
	public static void main(String[] args) {
		Random	random=new Random();
		int	n=random.nextInt(Integer.MAX_VALUE);
		int	pos=3; // 3 => 천의 자리 수 (10^3)
		int	d=
		System.out.println(n+" => "+d);
	}
}
```

19. (기수정렬: Radix sort, 십진수) 다음은 0 이상 최대 3자리까지 정수(int)들을 기수정렬하는 코드의 예이다. 아래 코드를 입력하고 실행하면서 기수정렬 구현법을 익히시오. (Reference: https://en.wikipedia.org/wiki/Radix_sort, CC-BY-SA, https://opendatastructures.org/ods-cpp/11_2_Counting_Sort_Radix_So.html CC BY 2.5 CA).

```
public class Test {
	public static void main(String[] args) {
		int	n[]={4,423,312,34,43,2,223,42};
		radixSort(n);
		System.out.println(Arrays.toString(n));
	}
	private static void radixSort(int[] n) { // 세자리 십진수 대상
		int	Radix=10;
		LinkedList<Integer>	queue[]=new LinkedList[Radix]; // 십진수 0~9 대응 큐 생성
		for (int q = 0; q < queue.length; q++) queue[q]=new LinkedList<>(); // 십진수 0~9 대응 큐 생성

		for (int i = 0; i < n.length; i++) queue[(n[i]/1)%Radix].add(n[i]); // 1의 자리 값 기준 큐 삽입
		for (int q = 0, i=0; q < queue.length; q++) while(!queue[q].isEmpty()) n[i++]=queue[q].remove(); // 큐 자료 재정렬

		for (int i = 0; i < n.length; i++) queue[(n[i]/10)%Radix].add(n[i]); // 10의 자리 값 기준 큐 삽입
		for (int q = 0, i=0; q < queue.length; q++) while(!queue[q].isEmpty()) n[i++]=queue[q].remove(); // 큐 자료 재정렬

		for (int i = 0; i < n.length; i++) queue[(n[i]/100)%Radix].add(n[i]); // 100의 자리 값 기준 큐 삽입
		for (int q = 0, i=0; q < queue.length; q++) while(!queue[q].isEmpty()) n[i++]=queue[q].remove(); // 큐 자료 재정렬
	}
}
```

20. (**실습**: 기수정렬, 십진수) 다음은 최대 9자리까지 십진수들을 기수정렬하는 코드의 일부분이다. 아래 코드를 완성하시오.

```
public class Test {
	public static void main(String[] args) {
		int	n[]={432109876,423,312,34,43,21,223,42};
		radixSort(n, 10, 9); // 최대 9자리까지 10진수 기수정렬
		System.out.println(Arrays.toString(n));
	}
	private static void radixSort(int[] n, int Radix, int Length) {
	}
}
```

21. (기수정렬: Radix sort using Counting sort, 십진수) 다음은 0 이상 최대 3자리까지 정수(int)들을 계수정렬을 사용하여 기수정렬하는 코드의 예이다. 아래 코드를 입력하고 실행하면서 계수정렬을 사용하는 기수정렬 구현법을 익히시오. (Reference: https://opendatastructures.org/ods-cpp/11_2_Counting_Sort_Radix_So.html CC BY 2.5 CA)

```
public class Test {
	public static void main(String[] args) {
		int	n[]={9,21,3,7,43,111,42,1,5,7};
		radixSort(n, 10, 3);
		System.out.println(Arrays.toString(n));
	}
	private static void radixSort(int[] n, int Radix, int Length) {
		for (int pos = 0, v=1; pos < Length; pos++, v*=Radix) {
			countingSort(n, v, Radix); // v: 배열 n의 정렬에 사용할 자리값 (예: 1, 10, 100)
		}
	}
	private static void countingSort(int[] n, int v, int Radix) {
		int	m[]=new int[n.length], count[]=new int[Radix];
		for (int i = 0; i < n.length; i++) count[(n[i]/v)%Radix]++;
		for (int i = 1; i < count.length; i++) count[i]+=count[i-1];
		for (int i = n.length-1; i >=0; i--) m[--count[(n[i]/v)%Radix]]=n[i];
		for (int i = 0; i < m.length; i++) n[i]=m[i];
	}
}
```

22. .

```
public class Test {
}
```

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.