1. (해싱, 선형조사법) 다음은 선형조사법으로 해싱을 구현한 코드이다. 아래 코드를 입력하고 실행하면서 해싱 구현법을 학습하시오. (Reference: http://opendatastructures.org/versions/edition-0.1e/ods-java/5_1_ChainedHashTable_Hashin.html, CC-BY-2.5-CA, Reference: https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/SeparateChainingHashST.java.html, GPLv3)

```
public class Test {
	public static void main(String[] args) {
		SimpleLinearProbingHashTable ht=new SimpleLinearProbingHashTable(1000);
		System.out.println(ht.put("Korea"));
		System.out.println(ht.put("Korea"));
		System.out.println(ht.put("Japan"));
		System.out.println(ht.get("Korea"));
		System.out.println(ht.get("Japan"));
		System.out.println(ht.get("China"));
	}
}

// Reference: http://opendatastructures.org/versions/edition-0.1e/ods-java/5_2_LinearHashTable_Linear_.html, CC-BY-2.5-CA
// Reference: https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/LinearProbingHashST.java.html, GPLv3
public class SimpleLinearProbingHashTable {
	private int HashTableSize;
	Object hashTable[];
	public SimpleLinearProbingHashTable(int size) {
		HashTableSize=size;
		hashTable=new Object[HashTableSize];
	}
	public boolean put(Object key) {
		int index=hash(key); // 삽입할 key 값을 해쉬테이블 인덱스로 변환
		while(hashTable[index]!=null){ // 현재 버켓이 사용 중이라면
		    if(hashTable[index].equals(key)) return false; // 삽입할 key 값 기삽입되어 있음
		    index=(index+1)%HashTableSize; // 다음 버켓 인덱스 계산
		}
		hashTable[index]=key; // 빈 버켓에 key 값 삽입
		return true; // 삽입 성공
	}
	private int hash(Object key) {
		return (key.hashCode()&0x7FFFFFFF)%HashTableSize;
	}
	public Object get(Object key) {
		int index=hash(key); // 탐색 key 값을 해쉬테이블 인덱스로 변환
		while(hashTable[index]!=null){ // 현재 버켓이 사용 중이라면
		    if(hashTable[index].equals(key)) return hashTable[index]; // 탐색 key 값 발견
		    index=(index+1)%HashTableSize; // 다음 버켓 인덱스 계산
		}
		return null; // 탐색 key 값이 존재하지 않음
	}
}
```

2. (해싱, 체인법) 다음은 체인법으로 해싱을 구현한 코드이다. 아래 코드를 입력하고 실행하면서 해싱 구현법을 학습하시오. (Reference: http://opendatastructures.org/versions/edition-0.1e/ods-java/5_1_ChainedHashTable_Hashin.html, CC-BY-2.5-CA, Reference: https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/SeparateChainingHashST.java.html, GPLv3)

```
public class Test {
	public static void main(String[] args) {
		SimpleChainHashTable ht=new SimpleChainHashTable(1000);
		System.out.println(ht.put("Korea"));
		System.out.println(ht.put("Korea"));
		System.out.println(ht.put("Japan"));
		System.out.println(ht.get("Korea"));
		System.out.println(ht.get("Japan"));
		System.out.println(ht.get("China"));
	}
}
// Reference: http://opendatastructures.org/versions/edition-0.1e/ods-java/5_1_ChainedHashTable_Hashin.html, CC-BY-2.5-CA
// Reference: https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/SeparateChainingHashST.java.html, GPLv3
public class SimpleChainHashTable {
	private int HashTableSize;
	LinkedList<Object> hashTable[];
	public SimpleChainHashTable(int size) {
		HashTableSize=size;
		hashTable=new LinkedList[HashTableSize];
		for (int i = 0; i < hashTable.length; i++) hashTable[i]=new LinkedList<>();
	}
	public boolean put(Object key) {
		if(get(key)!=null) return false; // 삽입할 key 값 기삽입되어 있음
		hashTable[hash(key)].add(key); // key 값을 연결리스트에 삽입
		return true;
	}
	private int hash(Object key) {
		return (key.hashCode()&0x7FFFFFFF)%HashTableSize;
	}
	public Object get(Object key) {
		for (Object v : hashTable[hash(key)]) if(v.equals(key)) return v;
		return null;
	}
}
```

3. (자바클래스 HashMap, HashSet) 다음은 해싱을 구현한 자바클래스 HashSet와 HashMap의 사용 예시 코드이다. 아래 코드를 입력하고 실행하면서 HashSet와 HashMap의 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};
		HashSet<Integer>  set=new HashSet<>();
		for (int i = 0; i < n.length; i++) set.add(n[i]); // 해시테이블에 자료 삽입
		System.out.println(set);
		System.out.println(set.size()); // 해시테이블 내 총 자료 개수 반환
		set.remove(20); // key 값 20 삭제
		System.out.println(set);
		System.out.println(set.contains(30)); // key 값 30이 존재하는 경우 true 반환
		System.out.println(set.contains(33)); // key 값 33이 존재하지 않는 경우 false 반환
		for (Integer key : set) {
			System.out.print(key+" ");
		}
	}
}
```

```
public class Test {
	public static void main(String[] args) {
		HashMap<String, Integer>	map=new HashMap<>();
		map.put("Korea", 32); // <key, value>가 <"Korea", 32>인 자료 삽입
		map.put("Japan", 50);
		map.put("France", 10);
		map.put("Mexico", 30);
		map.put("China", 16);
		System.out.println(map);
		map.put("Japan", 70); // key 값 "Japan"의 value를 70으로 변경
		System.out.println(map);
		map.remove("Japan"); // key 값 "Japan"에 해당하는 자료 삭제
		System.out.println(map);
		System.out.println(map.size()); // 해시테이블 내 총 자료 개수 반환
		System.out.println(map.containsKey("Korea")); // key "Korea" 존재 시 true 반환
		System.out.println(map.containsKey("Germany")); // key "Germany" 부재 시 false 반환
		System.out.println(map.get("Korea")); // key 값 "Korea"에 대응되는 value 반환
		System.out.println(map.get("Germany")); // key 값 부재 시 null 반환
		for (String key : map.keySet()) {
			System.out.println(key+"=>"+map.get(key));
		}
	}
}
```

4. (**실습**: 배열 내 최빈값 탐색) 다음은 배열 n에 저장된 0~100 범위의 정수 중 출현 빈도수가 최대인 정수(최빈값,mode)를 출력하는 코드이다. 예를 들어, {1,3,7,4,8,3,7,3,3,7}에서 최빈값은 3이다. 이 코드를 아래 각 실습 조건에 맞게 완성하시오. 또한 아래 각 방법의 시간복잡도는 얼마인가? (References: 프로그래밍대회에서 배우는 알고리즘 문제해결전략, 2012, 구종만 저, (p.93-94))

   - **실습** #1: HashMap을 이용하여 구현하시오. 배열 n의 각 정수 n[i]에 대해, n[i]를 HashMap의 key 값을 해석하여, n[i]가 HashMap에 존재하지 않는 경우 빈도수 1을 value로 설정하여 HashMap에 저장하고, n[i]가 HashMap에 존재하는 경우 기존 빈도수(HashMap 내 key 값 n[i]에 대응하는 value)의 1 증가된 값을 value로 설정하여 HashMap에 재저장하는 작업을 수행한다. 이후 HashMap 내 각 key 값을 순차 방문하면서 가장 큰 value를 갖는 키 값을 탐색한다.

   - **실습** #2: TreeMap을 이용하여 구현하시오. HashMap 대신 TreeMap을 사용하는 것을 제외하면 구현 방법은 HashMap의 경우와 동일하다.

   - **실습** #3: 배열 n을 정렬한 후, 정렬된 배열을 순차방문하면서 연속된 동일 값의 빈도수를 계산하여 그 빈도수가 최대인 값을 갱신하는 방식으로 구현하시오.

   - **실습** #4: 배열 n 내 정수 범위를 0~100으로 가정하고, 크기 101의 정수 배열 count를 만든 후, 배열 n 내 각 정수 n[i]에 대해, n[i]를 count 배열의 인덱스로 고려하여 count[n[i]]의 값을 1 증가시키는 작업을 반복한다. 이후 count 배열 내 가장 큰 값이 저장된 인덱스를 출력한다.

```
public class Test {
    public static void main(String[] args) {
            int     n[]={1,3,7,4,8,3,7,3,3,7};




    }
}
```

References

- C로 쓴 자료구조론 (Fundamentals of Data Structures in C, Horowitz et al.). 이석호 역. 사이텍미디어. 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.