1. 다음은 리스트에 77,99,88을 순차 저장 후 리스트 전체 자료를 출력하는 코드이다. 리스트에 int 형의 정수를 저장하기 위해 int 자료형의 wrapper class인 Integer를 사용하여 LinkedList 객체를 생성하였다. LinkedList<int>로 작성할 경우 오류 발생하므로 주의하시오.

```
public class Test {
	public static void main(String[] args) {
		LinkedList<Integer>	list=new LinkedList<>();
		list.add(77); // list.addLast(77)와 동일
		list.add(99);
		list.add(88);
		System.out.println(list); // 리스트 내 전체 자료 출력
	}
}
```

2. (**실습**) 1부터 100까지의 숫자들이 순차 저장된 리스트를 생성한 후 그 내용 전체를 출력하는 코드를 작성하시오.

3. (**실습**) 리스트에 "한국", "미국", "일본"을 순차 저장한 후 리스트 전체 자료를 출력하는 코드를 작성하시오. (참고: LinkedList<String> 사용)

4. 다음은 리스트에 77,99,88을 순차 저장 후 0번째, 1번째, 2번째 각 자료를 읽은 후 출력하는 코드이다. 이 코드를 입력하고 실행해 보시오.

```
public class Test {
	public static void main(String[] args) {
		LinkedList<Integer>	list=new LinkedList<>();
		list.add(77);
		list.add(99);
		list.add(88);
		int		v1=list.get(0);
		int		v2=list.get(1);
		int		v3=list.get(2);
		System.out.println(v1);
		System.out.println(v2);
		System.out.println(v3);
	}
}
```

5. (**실습**) 0~100까지의 범위의 임의 정수를 1000000(백만)개 생성하면서 생성된 순서대로 정수를 LinkedList에 저장한 후 리스트 내 99999번째 자료를 출력하는 코드를 작성하시오.

6. (배열 및 연결리스트 임의 접근 총 소요 시간 비교) 다음은 배열과 연결리스트에 이미 전체 자료가 저장되어 있다고 가정하고, 저장된 전체 자료(예: 10000개)를 임의 접근하는 총 소요 시간을 비교하는 코드이다. 아래 코드를 입력하고 실행하면서 배열이 갖는 상수 시간 임의 위치 접근의 장점과 리스트를 get(i)를 통해 접근하는 방식의 단점을 생각해 보시오.

```
public class Test {
	public static void main(String[] args) {
		int		N=10000; // 자료 개수

		int		s[]=new int[N]; // 배열 생성
		for (int i=0; i < N; i++) s[i]=i; // 배열 내 자료 저장

		LinkedList<Integer>	list=new LinkedList<>(); // 리스트 생성
		for (int i=0; i < N; i++) list.add(i); // 리스트 내 자료 삽입

		long		start;
		start=System.currentTimeMillis();
		for (int i = 0; i < N; i++){ int v=s[i]; }
		System.out.println((System.currentTimeMillis()-start)+" ms");

		start=System.currentTimeMillis();
		for (int i = 0; i < N; i++){ int v=list.get(i); }
		System.out.println((System.currentTimeMillis()-start)+" ms");
	}
}
```

7. 다음은 리스트에 77,99,88을 순차 저장하고, 1번째 자료를 삭제한 후 리스트 내용을 출력하는 코드이다. 이 코드를 입력하고 실행해 보시오.

```
public class Test {
	public static void main(String[] args) {
		LinkedList<Integer>	list=new LinkedList<>();
		list.add(77);
		list.add(99);
		list.add(88);
		list.remove(1);
		System.out.println(list);
	}
}
```

8. (**실습**) 1부터 100까지의 숫자들이 순차 저장된 리스트를 생성한 후 3번째 자료를 삭제한 다음 리스트 내용 전체를 출력하는 코드를 작성하시오.

9. 다음은 3,4,5,6,7이 순차 저장되어 있는 배열의 0번째 위치에 새로운 자료 2를 삽입하여 배열의 내용을 2,3,4,5,6,7로 변경하는 코드이다. 이 코드를 입력하고 실행해 보시오.

```
public class Test {
	public static void main(String[] args) {
		int	n[]={3,4,5,6,7,0,0,0,0,0};

		for (int i = 4; i >=0; i--) n[i+1]=n[i]; // n[0..4]를 n[1..5]로 이동
		n[0]=2; // n[0]에 2를 삽입
		System.out.println(Arrays.toString(n)); // [2,3,4,5,6,7,0,0,0,0]
	}
}
```

10. (**실습**) 아래 코드에서 배열 n에 저장된 0번째 자료를 삭제하여 배열의 내용이 다음과 같이 변경되도록 아래 코드를 완성하시오.

- 실행결과: [4, 5, 6, 7, 0, 0, 0, 0, 0, 0]

```
public class Test {
	public static void main(String[] args) {
		int	n[]={3,4,5,6,7,0,0,0,0,0};


		System.out.println(Arrays.toString(n));
	}
}
```

11. 다음은 리스트에 1,2,3을 순차 저장하고, 0번째 자료 위치에 7을 삽입하는 코드이다. 이 코드를 입력하고 실행해 보시오.

```
public class Test {
	public static void main(String[] args) {
		LinkedList<Integer>	list=new LinkedList<>();
		list.add(1); // list.addLast(1)와 동일
		list.add(2);
		list.add(3);
		list.addFirst(7);
		System.out.println(list);
	}
}
```

12. (배열 및 연결리스트 삽입 시간 비교) 다음은 10000개 자료를 배열과 연결리스트의 0번째 위치에 반복적으로 삽입하는 총 소요 시간을 비교하는 코드이다. 이 코드를 입력하고 실행해 보시오.

```
public class Test {
	public static void main(String[] args) {
		long	start;
		int	N=10000; // 자료 개수
		int	s[]=new int[N]; // 배열 생성
		LinkedList<Integer>	list=new LinkedList<>(); // 리스트 생성

		start=System.currentTimeMillis();
		for (int i=0; i < N; i++){
			for (int j = i-1; j >=0; j--) s[j+1]=s[j]; // 배열 자료 전체 오른쪽 한칸 이동
			s[0]=i; // 배열 0번째 위치에 자료 삽입
		}
		System.out.println((System.currentTimeMillis()-start)+" ms");

		start=System.currentTimeMillis();
		for (int i=0; i < N; i++) list.addFirst(i); // 연결리스트 0번째 위치에 자료 삽입
		System.out.println((System.currentTimeMillis()-start)+" ms");
	}
}
```

13. (**실습**) 다음은 기존 자료가 저장되어 있는 배열의 0번째 위치에 새로운 자료를 삽입하여 기존 배열을 크기 1 증가된 배열로 변경하는 코드이다. System.arraycopy(s, 0, t, 1, s.length)를 반복문으로 대체하여 작성하시오.

- System.arraycopy(s, 0, t, 1, s.length)는 s[i]를 t[j]에 대입하는 작업을 i=0,j=1부터 시작하여 s.length번 반복하는 함수이다.
- 삽입 전: [화, 수, 목, 금, 토, 일]
- 삽입 후: [월, 화, 수, 목, 금, 토, 일]

```
public class Test {
	public static void main(String[] args) {
		char	s[]= {'화','수','목','금','토','일'};
		System.out.println(Arrays.toString(s)); // 삽입 전
		char	t[]=new char[s.length+1];
		System.arraycopy(s, 0, t, 1, s.length);
		t[0]='월';
		s=t;
		System.out.println(Arrays.toString(s)); // 삽입 후
	}
}
```

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.