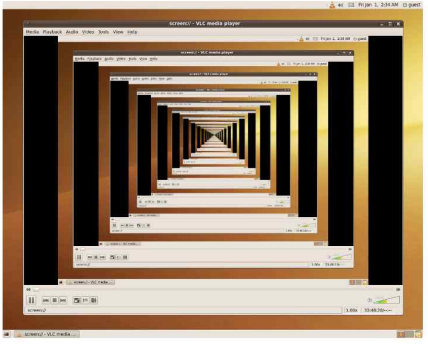
알고리즘 문제풀이

7.재귀함수

재귀함수

- 재귀함수(再歸函數, recursion)는 정의 단계에서 자신을 재참조하는 함수를 뜻한다.
- 하나의 함수에서 자신을 다시 호출하여 **스택구조**로 작업을 수행하는 방식으로 주어진 문제를 푸는 방법이다. 재귀 호출이나 되부름이라고도 한다.
- **피보나치수열, 팩토리얼, 트리의 깊이우선탐색 등은 재귀함수 형태로 구현가능.**
- 객체의 주소값 비교는 굳이 재귀함수를 사용할 이유가 없이 연산자만 사용하면 가능
- 아무런 조건이 없이 함수를 계속적으로 호출하기 때문에 무한 반복된다.
- 따라서, while 반복문과 같은 맥락으로 조건을 적용하여 무한 루프에 빠지지 않도록 주의하면서 코드를 작성해야 한다.

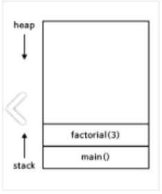
재귀는 마치.. 무의식 속의 무의식으로 들어갔다가 결국 현실로 되돌아오는 인셉션과 비슷하다.

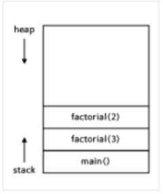


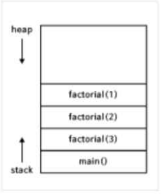
위키피디아

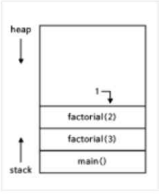
Stack Overflow는
Stack 영역의 메모리가 지정된 범위를 넘어갈 때 발생한다.
Stack 메모리는 보통 지역 변수가 저장되는 영역이다.

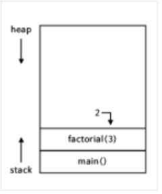
- jvm에서도 마찬가지로 메소드를 호출시 스택 메모리가 필요하다.
- 이때 약 320kb ~ 1mb 미만의 메모리를 사용한다.
- 즉 재귀호출시 깊이가 깊어짐에 따라 스택 메모리를 계속 할당되어야 한다.
- 결국 메모리 때문에 7~8천회이상 넘어가면 문제가 생길겁니다.

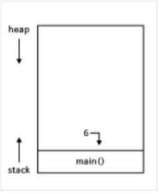

heap
factorial(3)
stack
main()


heap
factorial(2)
factorial(3)
stack
main()


heap
factorial(1)
factorial(2)
factorial(3)
stack
main()


heap
1
factorial(2)
factorial(3)
stack
main()


heap
2
factorial(3)
stack
main()


heap
6
stack
main()

재귀함수

```
public static void main(String[] args)  {
      recursiveTest(1);
}
```

```
결과:
count : 1
count : 2
count : 3
count : 4
count : 5
```

```
public void recursiveTest(int count)  {
   System.out.println("count : " + count);

   count++;

   if(count <= 5)       {
      recursiveTest(count);
   }
}
```

재귀함수

자연수 N이 입력되면 재귀함수를 이용하여 1부터 N까지를 출력하는 프로그램을 작성하세요.

▣ 입력설명

첫 번째 줄은 정수 N(3<=N<=10)이 입력된다.

▣ 출력설명

첫째 줄에 출력한다.

▣ 입력예제 1

3

▣ 출력예제 1

1 2 3

```
public class Main1_range {
	public static void main(String[] args) {
		Scanner s = new Scanner(System.in);
//		int len = s.nextInt();
		int len = 3;
		solution(len);
	}//end main

	private static void solution(int len) {
		if(len==0) return;
		else	{
//			System.out.print(len+" ");//push:3 2 1
			solution(len - 1 );
			System.out.print(len+" ");//pop:1 2 3
		}
	}
}//end class
```

재귀함수를 이용한 이진수 출력

10진수 N이 입력되면 2진수로 변환하여 출력하는 프로그램을 작성하세요. 단 재귀함수를 이용해서 출력해야 합니다.

▣ 입력설명

첫 번째 줄에 10진수 N(1<=N<=1,000)이 주어집니다.

▣ 출력설명

첫 번째 줄에 이진수를 출력하세요.

▣ 입력예제 1

11

▣ 출력예제 1

1011

```
public class Main2_binary {
	public static void main(String[] args) {
		int len = 11;
		solution(len);
	}//end main
	private static void solution(int len) {
		if(len==0) return;
		else	{
			System.out.print(len+" ");  //11 5 2 1 : stack push 순서
			solution(len/2);

			//stack pop 순서 : 1%2=1, 2%2=0, 5%2=1, 11%2=1
			System.out.print(len%2+" ");  //1011 : 11
		}
	}
}//end class
```

팩토리얼

자연수 N이 입력되면 N!를 구하는 프로그램을 작성하세요.
예를 들어 5! = 5*4*3*2*1=120입니다.

▣ 입력설명
첫 번째 줄에 자연수 N(1<=N<=100)이 주어집니다.

▣ 출력설명
첫 번째 줄에 N팩토리얼 값을 출력합니다.

▣ 입력예제 1
5

▣ 출력예제 1
120

```
public class Main3_factorial {
	public static void main(String[] args) {
		int len = 5;
		//팩토리얼 결과를 최종 리턴받아 출력 : 120
		System.out.println(solution(len));
	}//end main

	private static int solution(int len) {
		//5 4 3 2 1 : stack push 순서
		System.out.print(len+" ");
		if(len==1) return 1;
		else	{
			//stack pop 순서 : 1*2*3*4*5
			return solution(len-1)*len;
		}
	}
}//end class
```

피보나치 수열

1) 피보나키 수열을 출력한다. 피보나치 수열이란 앞의 2개의 수를 합하여 다음 숫자가 되는 수열이다.
2) 입력은 피보나치 수열의 총 항의 수 이다. 만약 7이 입력되면 1 1 2 3 5 8 13을 출력하면 된다.

▣ 입력설명

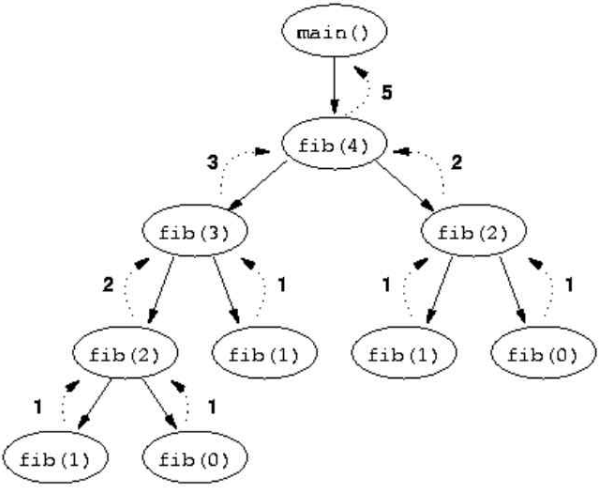
첫 줄에 총 항수 N(3<=N<=45)이 입력된다.

▣ 출력설명

첫 줄에 피보나치 수열을 출력합니다.

▣ 입력예제 1

10


main()
5
fib(4)
3
2
fib(3)
fib(2)
2
1
1
1
fib(2)
fib(1)
fib(1)
fib(0)
1
1
fib(1)
fib(0)

```
public class Main4_fibonacci {
	public static void main(String[] args) {
		int len = 10;
		//1 1 2 3 5 8 13 21 34 55
		for (int i = 1; i <= len; i++) {
			System.out.print(solution( i ) + " ");
		}
	}//end main

	private static int solution(int i) {
		if(i<=1) return i;
		else	{
			return solution(i-2) + solution(i-1);
		}
	}
}//end class
```