2014 자료구조및실습 실습과제 3

1. 반복적인 방법과 순환적인 방법으로 거듭제곱 함수를 구현한다.

(1) 함수 형태

- 반복적인 방법: double hkdPowerIter(double x, int n)
- 순환적인 방법: double hkdPowerRecur(double x, int n)

(2) 다양한 n에 대해 실제 실행시간을 추출해 본다. (지난 실습문제 참조)

(3) 강의자료 13쪽의 표와 가장 유사한 결과를 만들어보자. 이때, 실제 수행 속도를 측정하여 표를 만든다.

2. 반복적인 방법과 순환적인 방법으로 피보나치 수열을 구현한다.

(1) 함수 형태

- 반복적인 방법: hkdFiboIter (int n) // n번째 수를 구하는 함수
- 순환적인 방법: hkdFiboRecur (int n) // n번째 수를 구하는 함수

(2) 두 함수의 결과가 같음을 확인하라.

- n을 1부터 하나씩 증가, 오버플로우가 나타날 때 까지 출력

(3) 재귀적인 방법으로 호출하였을 때 함수가 중복되어 호출되는 것을 확인할 수 있도록 재귀호출 함수 앞부분에 "Entering Fibo(n)"코드를 넣고, n=10을 넣었을 때 각 함수가 호출된 횟수를 계산해 다음과 같이 출력하시오.

```
Fibo(10)        = 1번
Fibo(9)         = ??번
...
Fibo(0)         = ??번
```

※ 단, 이것은 자동으로 해도 되고(프로그램으로), 출력된 결과를 찾아 수동으로 해도 됨

2. 다음과 같은 트리 모양을 출력하는 함수를 작성하여 보자.

```
------------------×------------------
---------×------------------×----------
----×---------×---------×---------×-----
-×----×----×----×----×----×----×-----×--
```

(1) 위와 같은 모양을 출력하는 순환 함수 draw_tree(int row, int left, int right)를 설계하여 보자.

- 먼저 함수의 매개 변수는 row과 left, right가 된다. row은 ×를 그리는 행을 표시한다.
- 가장 위에 있는 행이 0이고 아래로 내려갈수록 숫자가 증가한다고 생각하자.
- left와 right는 각각 주어진 영역의 왼쪽 끝과 오른쪽 끝을 나타낸다.
- draw_tree 함수는 주어진 행에서 주어진 영역의 중간 위치를 계산하고 중간 위에'×'를 출력한 다음에 주어진 영역을 2개로 나누어 각각의 영역에 대하여 각각 draw_tree 함수를 순환 호출하면 된다.
- 영역이 너무 작으면. 예를 들어 (right-left) < 3이면 그냥 복귀하면 된다.
- 구현을 쉽게 하기 위하여 다음과 같이 2차원 문자열 배열에 그린 다음, 한꺼번에 화면에 출력하도록 하자.

```
#define MAX_HEIGHT 50
#define MAX_WIDTH 40
char screen[MAX_HEIGHT][MAX_WIDTH];

void draw_tree(int row, int left, int right)
{
	int mid;
	if( (right-left)<3 ){
		return;
	}
	mid = (right+left)/2;
	screen[row][mid] = 'X';
	draw_tree(____, ____, ____); // 왼쪽 영역
	draw_tree(____, ____, ____); // 오른쪽 영역
}
```

(2) main 함수와 screen 문자열을 초기화시키는 init 함수, screen 문자열을 화면에 출력하는 함수 display()를 작성하여 프로그램을 수행시켜본다.

(3) 각 함수가 호출될 때마다 함수의 이름과 매개 변수를 출력하는 문장을 삽입하여 순환 호출이 어떻게 일어나는지를 살핀다.