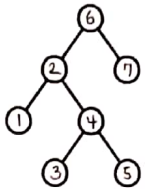
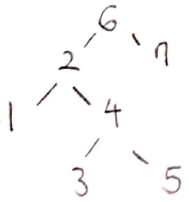
Written Homework 2

Due: 10월 26일 오후 9시

In class, we discussed the *height* of a binary search tree is an important measure of its efficiency of searching. This homework deals with a few related questions. **No late submission is accepted for this homework. Make sure to submit it correctly!**

1. (a) Fill in the nodes of the following binary tree with seven key values, 1, 2, 3, 4, 5, 6, and 7, so that the tree becomes a binary search tree.





   (b) What is the number of binary search trees with the keys, 1, 2, 3, 4, 5, 6, and 7, whose root node contains 4? 25

2. Consider the following two functions:

```
f(t)
{
  if(t=null) return 0
  l=f(LLINK(t))
  r=f(RLINK(t))
  if(l<r) return r+1
  else return l+1
}
```

```
g(t)
{
  if(t=null) return 0
  l=g(LLINK(t))
  r=g(RLINK(t))
  return l+r+1
}
```

Given a binary tree represented as linked nodes and the pointer `t` to its root, what do `f(t)` and `g(t)` return? f(t) : height + 1 (1-(a)의 경우 +) g(t) : 전체 노드의 수 (1-(a)의 경우 ?)

3. Suppose that 1023 keys $\{4k+1 \mid k = 1, 2, \ldots, 1023\} = \{5, 9, 13, \ldots, 4093\}$ are inserted into a binary search tree. Each of the 1023 keys is inserted once and only once. But we do not know the order of the insertions.

   (a) What is the maximum possible height of the resulting binary search tree? 1022

   (b) Consider a sequence of the insertions of the keys that results in a binary search tree of the maximum height. List the first five keys. (There can be many answers.) 5, 9, 13, 17, 21

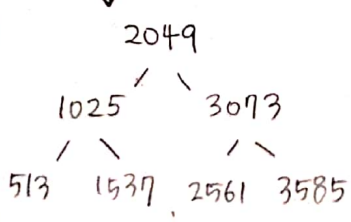
   (c) What is the minimum possible height of the resulting binary search tree? 9

   (d) 10 Suppose that a sequence of the insertions of the keys resulted in a binary search tree of the minimum height. List the first five keys that were inserted. (There is only one answer.) 2049, 1025, 3073, 513, 2561

   (e) What are the minimum and maximum possible total numbers of key comparisons it takes to insert all the 1023 keys? Hint: Note that $1023 = 2^{10} - 1$. Use the formula $\sum_{k=1}^{n} k \cdot 2^k = (n-1)2^{n+1} + 2$.

minimum : 8194

maximum : 522753



4. Suppose that n keys $\{1, 2, \ldots, n\}$ are inserted into a binary search tree. Each of the 1023 keys is inserted once and only once, and, therefore, there are $n!$ possible ways to insert the n keys. Recall that there are C_n different binary trees. (Recall WHW1.)

(a) Consider the first a few entries of C_n and $n!$ as follows:

n	C_n	$n!$
0	1	1
1	1	1
2	2	2
3	5	6
4	14	24
5	42	120
6	132	720
7	429	5040
8	1430	40320
9	4862	362880
10	16796	3628800

Prove that $C_n < n!$, for all $n \geq 3$. (Use mathematical induction.)

(b) Prove that each of all possible C_n binary trees arises from an insertion order, among $n!$ possible ways, of the above n keys. (Again, use mathematical induction!)

5. Use the code for HW5, together with properly completed add(), and write a program to perform the following experiment that estimates the average height of BST, for $N = 50, 100, 200$ and 500:

(a) Generate N random data using the standard C library function rand().

(b) Insert the generated data into a BST, and obtain the height of the resulting BST, whose height must be between $\lg N$ and $N - 1$.

(c) Repeat Step (b) 100 times. And calculate the average of 100 results.

Present your program and discuss your experiment result.

(a) $C_n = \frac{1}{n+1} \cdot \binom{2n}{n}$ (카탈란수) 이고, $n=3$인 경우 $C_3=5$, $3!=6$으로, $C_n < n!$ 을 만족한다. ($k \geq 3$) $C_n < n!$을 만족하는 $n=k$가 있다고 가정하자. $\frac{1}{k+1} {}_{2k}C_k < k!$은 성립하게 되고, $n=k+1$일 경우를 살펴보면,

A $\frac{1}{k+2} \cdot {}_{2(k+1)}C_{k+1} < (k+1)!$ 이 된다.

$\frac{1}{k+1} {}_{2k}C_k \times \left(\frac{k+1}{k+2} \cdot \frac{(2k+2)(2k+1)}{(k+1)(k+1)} \right)$

$= \frac{1}{k+2} \cdot {}_{2(k+1)}C_{k+1}$ 이고, 만약 $k! \times \left(\frac{2(2k+1)}{k+2} \right)$ 이 $(k+1)!$ 보다 작거나 같다면 A식은 성립하게 된다.

$k! \times \frac{4k+2}{k+2} \leq (k+1)!$

$\frac{4k+2}{k+2} \leq k+1$. $k \geq 3$인 수에서 $k+1$은 항상 4보다 크고, $\frac{4k+2}{k+2}$는 항상 4보다 작은 수이므로, 부등식은 성립하고, 이에 따라 A식 역시 성립한다.

따라서 모든 $n \geq 3$ 에서, $C_n < n!$은 성립하게 된다.

(b) $n=0$인 경우, $C_0=1$, $0!=1$ 이고, 문제의 조건을 만족한다.
$n=1$인 경우, $C_1=1$, $1!=1$이고, 문제의 조건을 만족한다.
($k \geq 1$)
$n=k$인 경우, 왼쪽 subtree의 노드개수와 오른쪽 subtree의 노드개수의 가능한 순서쌍은, $(k-1, 0)$ $(k-2, 1)$ $\cdots$ $(1, k-2)$ $(0, k-1)$ 이다. 이때 문제의 조건을 만족한다 하자. 즉, $n=k-1$, $n=k-2$, $\cdots$ $n=0$ 모두 문제의 조건을 만족하는 것이다. 이때,

$n=k+1$인 경우를 살펴보면, 마찬가지로 순서쌍을 나열한다면 $(k, 0)$, $(k-1, 1)$, $(k-2, 2) \cdots (2, k-2)$ $(1, k-1)$, $(0, k)$가 된다. $n=k$인 경우와, $n=k-1$, $n=k-2$, $\cdots$ $n=0$ 모두 문제의 조건을 만족하므로, 결국 $n=k+1$ 또한 문제의 조건을 만족한다고 말할 수 있다

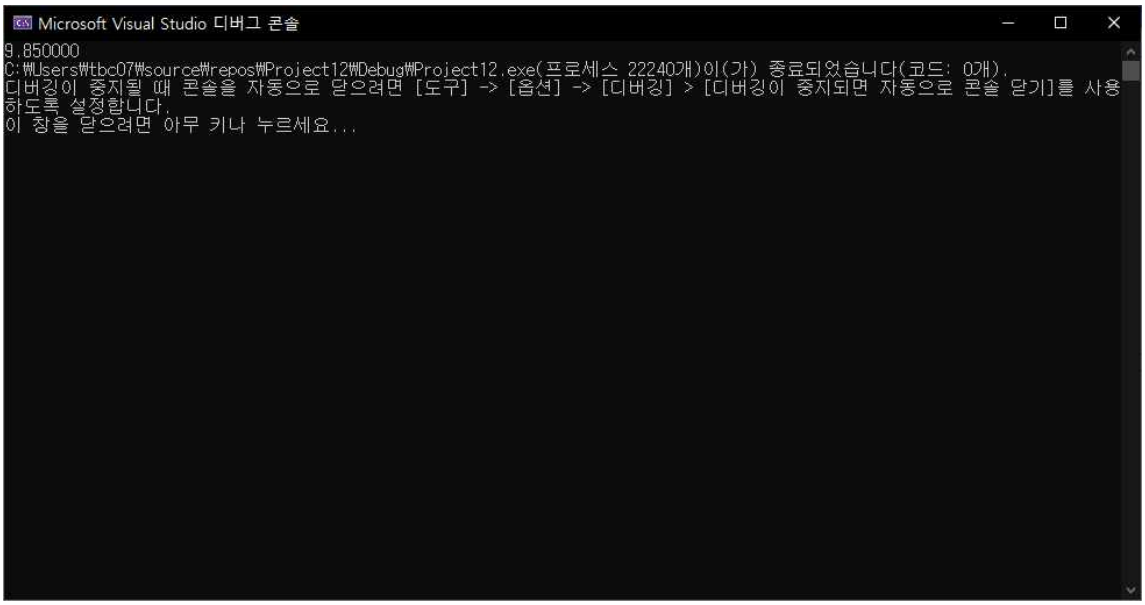
따라서, 위 식에 모든 n 값에 대해 문제의 조건이 성립함을 증명할 수 있다.

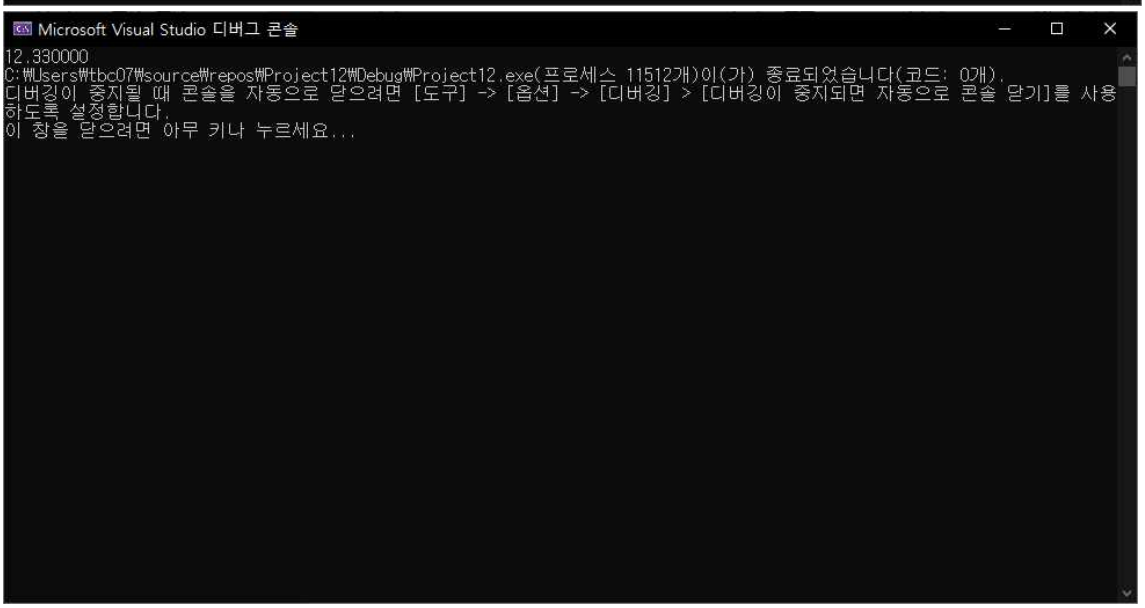
(a) main() 함수에서 N개의 랜덤 정수를 뽑아 배열 arr에 저장하였다.
(b) arr에 저장된 N개의 정수를 또다시 랜덤한 순서로 add() 함수에 넘겨, BST를 작성하였다. 그리고 BST의 height를 계산하여 배열 hArray에 저장하였다.
(c) (b)를 100번 반복하며, 매번 height를 계산한 후 main() 함수 마지막 부분에서 the average height of BST를 계산하였다.

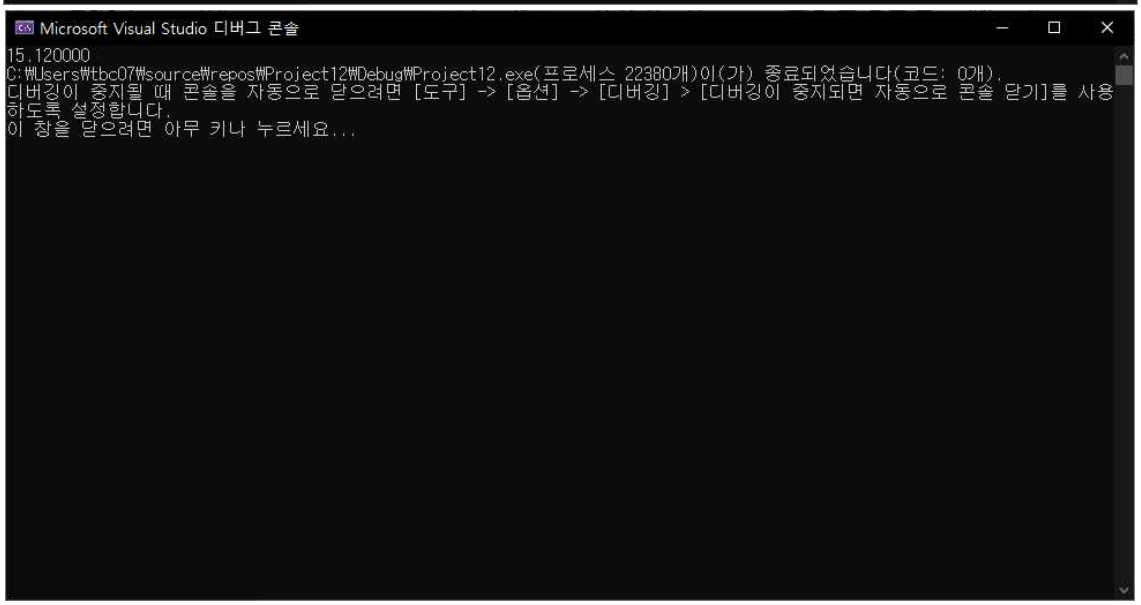
- 다음은 main()함수의 일부이다.

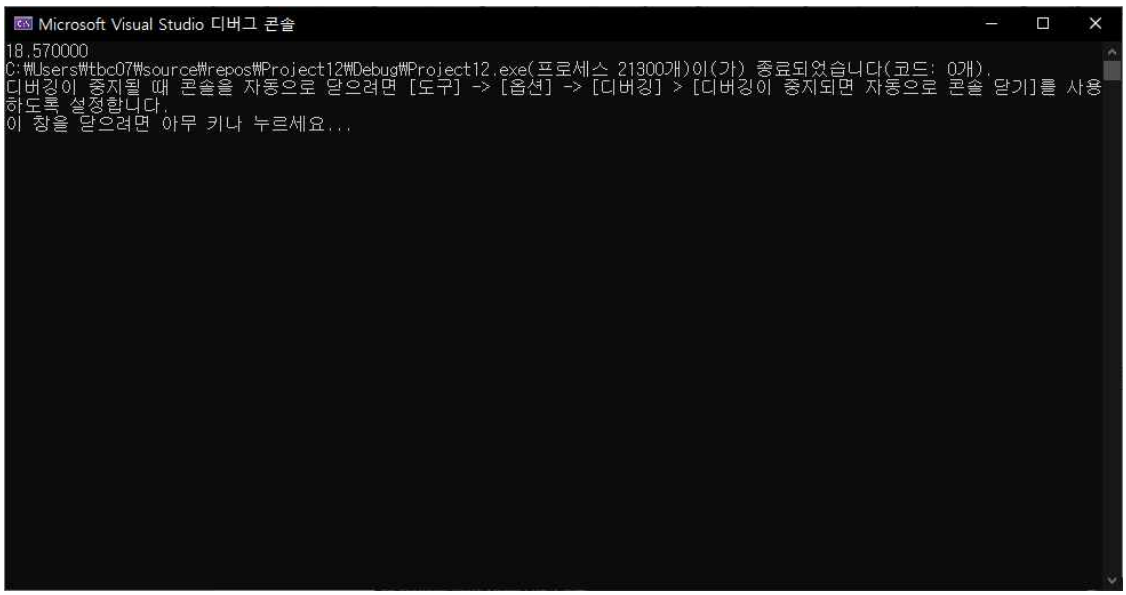
```
    srand((unsigned)time(NULL));
    init_pool();

    for (i = 0; i < N; i++) {
        arr[i] = rand();
    }

    for (k = 0; k < 100; k++) {
        for (i = 0; i < N;) {
            for (j = 0; j < N;) {
                num = rand() % N + 1;
                if (num != arr2[j]) {
                    arr2[j] = num;
                    j++;
                    add(arr[num - 1]);
                    i++;
                }
            }
        }

        hArray[k] = height(data);
        sum += hArray[k];
        data += N;
    }

    ave = (double)sum / 100;
    printf("%f", ave);
```

N=50일 경우, height의 평균은 9.85 였다.
N=100일 경우, height의 평균은 12.33 였다.
N=200일 경우, height의 평균은 15.12 였다.
N=500일 경우, height의 평균은 18.57 였다.


Microsoft Visual Studio 디버그 콘솔
9.850000
C:₩Users₩tbc07₩source₩repos₩Project12₩Debug₩Project12.exe(프로세스 22240개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...


Microsoft Visual Studio 디버그 콘솔
12.330000
C:₩Users₩tbc07₩source₩repos₩Project12₩Debug₩Project12.exe(프로세스 11512개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...


Microsoft Visual Studio 디버그 콘솔
15.120000
C:₩Users₩tbc07₩source₩repos₩Project12₩Debug₩Project12.exe(프로세스 22380개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...



코드 전체는 다음과 같다.
main() 함수의 N에 각각 50, 100, 200, 500을 넣어 실행하였다.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#define POOL_SIZE 500000

// record structure
struct record {
    int number;
    struct record* left;
    struct record* right;
};

// pool of memory
static struct record pool[POOL_SIZE]; // static because pool is strictly private
struct record* top = pool;  // a pointer variable for pool stack top.

// data
struct record* data = NULL; // Initially NULL.

void init_pool() // Initialize the pool; Use right instead of next!!!
{
    int i;
```

```
    struct record* r = pool;
    struct record* s;

    pool[POOL_SIZE - 1].right = NULL;

    for (i = 1; i < POOL_SIZE; i++) {
        s = r++;
        s->right = r;
    }
}

struct record* new_node()
{
    struct record* r;

    if (top == NULL)
        return NULL;

    r = top;
    top = r->right;  // Again, right instead of next.
    return r;
}

void add(int number)
{
    struct record* r = new_node();
    struct record* p = data;
    struct record* q;

    if (r == NULL) {
        printf("Can't add.  The pool is empty!\n");
        return;
    }
    else {
        r->number = number;
        r->left = NULL;
        r->right = NULL;

        if (p == NULL) {
            data = r;
            return;
        }
        else {
```

```
            while (p != NULL) {
                if (number <= p->number) {
                    q = p;
                    p = p->left;
                    if (p == NULL) {
                        q->left = r;
                        return;
                    }
                }
                else {
                    q = p;
                    p = p->right;
                    if (p == NULL) {
                        q->right = r;
                        return;
                    }
                }
            }
        }
    }
}

// returns the height of the BST.
int height(struct record* t)
{
    int max;

    if (t == NULL)
        return -1;
    else {
        if (height(t->left) >= height(t->right))
            max = height(t->left);
        else max = height(t->right);
        return (1 + max);
    }
}

int main() {
    int i;
    int j;
    int k;
    int arr[500];
    int arr2[500] = { 0 };
```

```
    int hArray[100];
    int num;
    int sum = 0;
    double ave;

    srand((unsigned)time(NULL));
    init_pool();

    for (i = 0; i < N; i++) {
        arr[i] = rand();
    }

    for (k = 0; k < 100; k++) {
        for (i = 0; i < N;) {
            for (j = 0; j < N;) {
                num = rand() % N + 1;
                if (num != arr2[j]) {
                    arr2[j] = num;
                    j++;
                    add(arr[num - 1]);
                    i++;
                }
            }
        }

        hArray[k] = height(data);
        sum += hArray[k];
        data = NULL;
    }

    ave = (double)sum / 100;
    printf("%f", ave);

    return 0;
}
```