풀이시간은 30 분이며 사정에 따라 최대 10 분간 추가 시간을 줄 수 있습니다. 문제 풀이를 마친 학생은 답안지를 제출하시기 바랍니다. 답안지를 제출한 후에는 퇴실할 수 있습니다.

Name: ______________ Student ID: ______________ Class: ______________

담당교수: 김종규

---

1. (10 points) 다음에 대하여 참, 거짓을 판별하고 그 이유를 설명하시오.
   (a) $2^{2+n} = O(2^n)$

      Answer:

      참. $2^{2+n} = 4 \times 2^n \leq c2^n$ where $c = 5$ for $n > 1$

   (b) $2^{2n} = O(2^n)$

      Answer:

      거짓. $2^{2n} = 2^n \times 2^n$.

      Assume that there is c such that $2^n \times 2^n \leq c2^n$. This implies that $2^n \leq c$, which is not possible to satisfy for all $n > 0$.

2. (10 points) 다음 그림 1은 배열 A[1..n] = {27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0} 을 binary tree 로 표현한 것이다. 여기서 A[3] 이 가리키는 subtree 가 heap 의 요건을 만족시키지 않기 때문에 그림 2 와 같이 정의된 알고리즘 Max-Heapify(A,3) 을 호출하였다. 그림 1 를 참고하여 Max-Heapify 가 호출될 때의 변화를 그림으로 설명하시오.

   Answer:

3. (10 points) 배열로 정의된 Heap 은 주어진 index i 에 대하여 `A[parent(i)] > A[i]` 인 관계가 성립한다. 여기서 parent 는 다음과 같다.

```
def parent(i):
  return i//2
```

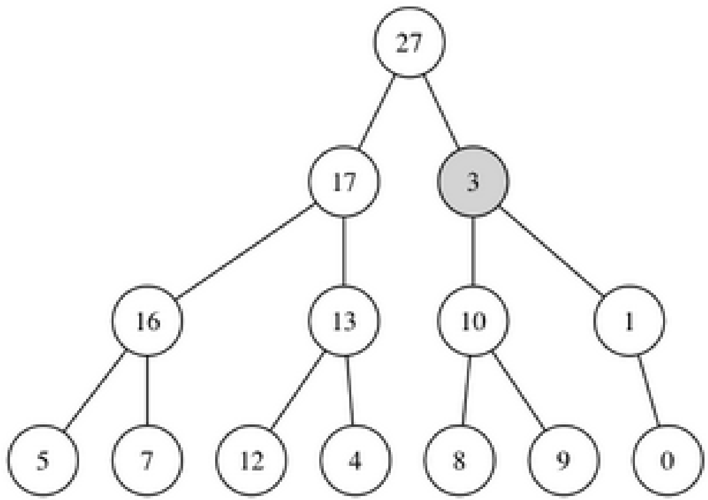

그림 1: Heapify input

다음과 같이 역순으로 정렬된 배열 A[1..n] 이 있다고 할 때

```
A[1] = 8
A[2] = 7
A[3] = -1
A[4] = -3
A[5] = -5
...
```

이 배열이 heap 의 조건을 만족하는가? 참, 거짓을 판별하고 그 이유를 설명하시오.

Answer:
참. parent 의 정의로 부터 left 와 right 는 다음과 같이 정의된다.

```
def left(n):
    return 2*n
def right(n):
    return 2*n+1
```

Heap 의 정의는 `A[i] > A[left(i)]` and `A[i] > A[right(i)]`
어떤 index i 에서 left(i) 와 right(i) 를 구하면 다음 관계가 성립한다.
$i <$ left(i) and $i <$ right(i)
주어진 배열이 역순으로 정렬되어 있으므로

```
MAX-HEAPIFY(A, i)
l = LEFT(i)
r = RIGHT(i)
if l ≤ A.heap-size and A[l] > A[i]
    largest = l
else largest = i
if r ≤ A.heap-size and A[r] > A[largest]
    largest = r
if largest ≠ i
    exchange A[i] with A[largest]
    MAX-HEAPIFY(A, largest)
```

그림 2: Max-Heapify

`A[i] > A[left(i)]` and `A[i] > A[right(i)]`
따라서 heap 의 정의를 만족한다.

4. (10 points) 다음 그림 5 은 binary tree 의 한 예를 보여주고 있다. 어떤 node `x` 가 left, right, parent node 에 대한 포인터와 key 에 대한 속성을 각각 `x.left`, `x.right`, `x.p`, `x.key` 로 나타낸다고 할 때, 다음 물음에 답하시오.

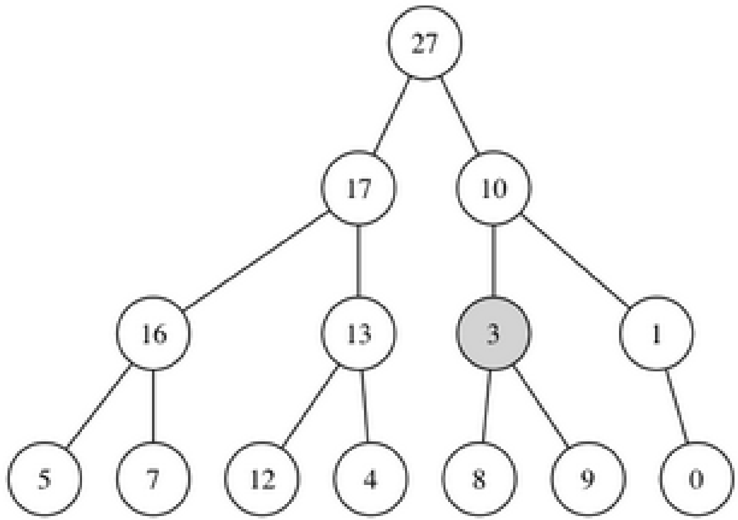

그림 3: Heapify snapshot 1

(a) 주어진 tree 의 높이를 계산하도록 다음 알고리즘을 완성하시오. 단 root node 만으로 이루어진 binary tree 의 높이는 1 로 정의한다.

```
def bt_height(x):
    if x == NIL:
        return 0
    lh = bt_height(x.left)
    rh = bt_height(x.right)
    return _______________
```

Answer:

```
def bt_height(x):
    if x == NIL:
        return 0
    lh = bt_height(x.left)
    rh = bt_height(x.right)
    return max(lh, rh) + 1
```

(b) 어떤 binary tree 의 높이가 h 라고 할 때 이 트리에 들어갈 수 있는 node 의 최대 수 $n(h)$ 는 얼마인가? 그 이유를 간단히 설명하시오.

Answer:

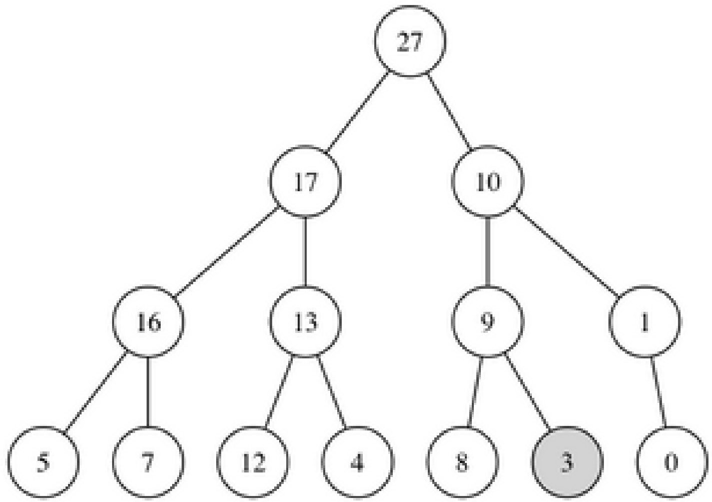

그림 4: Heapify snapshot 2

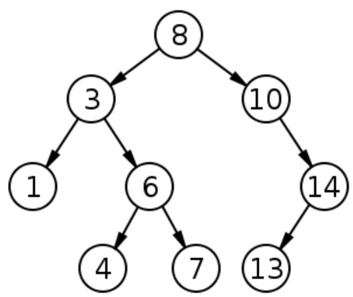

그림 5: An example of binary search tree

$2^h - 1$

$n(1) = 1 = 2^h - 1$

$n(2) = 3 = 2^h - 1$

$n(3) = 7 = 2^h - 1$

......

여기까지 작성하면 정답으로 인정

엄밀한 증명 (pf by induction):

$h = 1$ 일 때 성립

$h = k$ 일 때 성립한다고 가정

$h = k + 1$ 일 때 2^k 개의 노드가 추가됨

$n(k) + 2^k = 2^k - 1 + 2^k = 2^k(1 + 1) - 1 = n(k + 1)$

$\therefore n(h) = 2^h - 1$