

# Group Activity 01

(3인 혹은 4인으로 팀을 구성하여 아래의 문제를 푼다. 팀 구성은 매 시간마다 달라져도 된다.)

팀원1: \_\_\_\_\_

팀원2: \_\_\_\_\_

팀원3: \_\_\_\_\_

팀원4: \_\_\_\_\_

1. 다음의 각각의 함수의 최악의 경우의 시간복잡도를 점근적(asymptotic) 표기법으로 나타내면? 이유는?

```
int fun1(int n, int data[]) {  
    int sum = 0;  
    for (int i = 0; i < n; i+=2)  
        sum += data[i];  
    return sum;  
}
```

```
int fun2(int m, int A[], int n, int B[], int C[]) {  
    int i = 0, j = 0, k = 0;  
    while(i < m && j < n) {  
        if (A[i] < B[j])  
            C[k++] = A[i], i++;  
        else if (A[i] > B[j])  
            C[k++] = B[j], j++;  
        else  
            C[k++] = A[i], i++, j++;  
    }  
    while (i < m)  
        C[k++] = A[i], i++;  
    while (j < n)  
        C[k++] = B[j], j++;  
    return k;  
}
```

```
double fun3( int n ) {  
    double sum = 0;  
    for (double i = 1.0; i < n; i *= 1.5)  
        sum += i;  
    return sum;  
}
```

```

/* 단 max와 배열 data에 저장된 값은 모두 양수 */
int fun4(int n, int target, int data[] ) {
    int count = 0, i = 0, j = n-1;
    while (i<j) {
        if (data[i] + data[j] == target)
            count++;
    }
    return count;
}

```

```

/* 배열 data에 n개의 정수들이 오름차순으로 정렬되어 있음 */
int fun5( int n, int K, int data[] ) {
    int count = 0;
    for (int i=0; i<n; i++) {
        int result = binary_search(n, data, K-data[i]); /* 이진검색을 수행한다. */
        if (result != -1)
            count++;
    }
    return count;
}

```

```

/* 배열 A와 B에 각각 m개와 n개의 정수가 오름차순으로 정렬되어 저장. 그 외에 어떤 가정도 없음 */
void fun6(int m, int A[], int n, int B[]) {
    for (int i=0; i<m; i++) {
        for (int j=1; j<n; j*=2) {
            if (A[i] <= B[j])
                break;
        }
    }
}

```

```
int fun7(int n). {
    int count = 0;
    for (int i = n; i > 0; i /= 2)
        for (int j = 0; j < i; j++)
            count += 1;
    return count;
}
```

2. 스택과 FIFO 큐를 각각 배열과 단방향 연결리스트로 구현할 때 **push, pop, enqueue, dequeue** 등의 기본 연산의 시간복잡도는? 단, 배열로 구현할 경우 배열 재할당(array reallocation)에 소모되는 시간은 포함하지 않고 계산한다.

자료구조		Push / Enqueue	Pop / Dequeue
스택	배열		
	단방향 연결리스트		
FIFO 큐	원형(circular) 배열		
	단방향 연결리스트		

3. 프로그램에서 리스트(list)를 표현하는 대표적인 2가지 방법은 배열과 연결리스트이다. 또한 각각의 경우 데이터들을 크기순으로 정렬해서 저장할 수도 있고 그렇지 않을 수도 있으므로 총 4가지 방법을 생각해 볼수 있다. 길이가  $N$ 인 리스트를 이러한 4가지 방법으로 저장했을 때 다음 각각의 연산의 최악의 경우 시간복잡도는? 이유는?

- 검색: 어떤 값이 리스트에 포함되어 있는지 검사한다.
- 삽입: 새로운 하나의 값을 리스트에 추가한다.
- 삭제: 어떤 값을 리스트로 부터 삭제한다 (단 리스트에서 삭제할 값의 위치를 찾는데 걸린 시간은 제외한다.)

자료구조		검색(search)	삽입(insert)	삭제(remove)
배열	정렬 안함			
	오름차순으로 정렬함			
연결리스트	정렬 안함			
	오름차순으로 정렬함			

4. 다음 중  $O(n^2)$ 이 아닌 것은? 이유는?

- (1)  $15^{10}n + 12099$
- (2)  $n^{1.98}$
- (3)  $n^3/\sqrt{n}$

(4)  $2^{20}n$

5. 다음의 4가지 함수를 점근적 차수가 낮은 것부터 높은 것 순으로 나열하면? 이유는?

$$f_1(n) = 2^n \quad f_2(n) = n^{3/2} \quad f_3(n) = n \log^2 n$$

6. 다음 중 옳은 것을 모두 고르면? 이유는?

(1)  $(n+k)^m = \Theta(n^m)$ , 단 여기서  $k$ 와  $m$ 은 상수이다.

(2)  $2^{n+1} = O(2^n)$

(3)  $2^{2n+1} = O(2^n)$

7. 다음 중 옳은 것을 모두 찾아라.  $c$ 와  $d$ 는 양의 상수이고,  $c \leq d$ 이다. 이유를 설명하라.

(1)  $f(n) = O(n^c)$ 이고  $g(n) = O(n^d)$ 이면  $f(n) + g(n) = O(n^d)$  이다.

(2)  $f(n) = O(n^c)$ 이고  $g(n) = \Theta(n^d)$ 이면  $f(n) + g(n) = \Theta(n^d)$ 이다.

(3)  $f(n) = O(n^c)$ 이고  $g(n) = O(n^d)$ 이면  $f(n) + g(n) = \Theta(n^d)$ 이다.

(4)  $f(n) = \Theta(n^c)$ 이고  $g(n) = \Theta(n^d)$ 이면  $f(n) + g(n) = O(n^d)$ 이다.