

Object Oriented Programming (Week 4)

2023

KWANGWOON UNIVERSITY
DEPT. OF COMPUTER ENGINEERING

Contents

- Assignment 1-3. 1
- Assignment 1-3. 2
- Assignment 1-3. 3
- Assignment 1-3. 4

ASSIGNMENT 1-3. 1

Assignment 1-3. 1

- Write a program to perform the following matrix transformation **taking a three-dimensional coordinate as an input**. The operator **T** can be expressed as the composition.

$$T = T_3 \cdot T_2 \cdot T_1$$

where **T_1** is the rotation about the z-axis, **T_2** is the reflection about the yz-plane, and **T_3** is the orthogonal projection on the xy-plane. The standard matrices for these operators are

$$T_1 = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, T_2 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, T_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Thus, the standard matrix for **T** is

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -\cos \theta & \sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Input	Output
Degrees: 45 Coordinate: 1 1 1	0 1 0

Assignment 1-3. 1

▪ Matrix Multiplication

$$\textcircled{1} \quad T_2 \cdot T_1 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix}$$

$$a_1 = -1 * \cos \theta + 0 * \sin \theta + 0 * 0$$

$$\textcircled{2} \quad T_2 \cdot T_1 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -\cos \theta & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix}$$

$$a_2 = -1 * -\sin \theta + 0 * \cos \theta + 0 * 0$$

⋮

$$\textcircled{5} \quad T_2 \cdot T_1 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -\cos \theta & \sin \theta & 0 \\ \sin \theta & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix}$$

$$a_5 = 0 * -\sin \theta + 1 * \cos \theta + 0 * 0$$

⋮

$$\textbf{Result} \quad T_2 \cdot T_1 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -\cos \theta & \sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Assignment 1-3. 1

- `cos()`

- Returns the **cosine** of an angle of **x radians**.
- format

```
#include <cmath>
double cos (double x);
float cos (float x);
long double cos (long double x);
```

- `sin()`

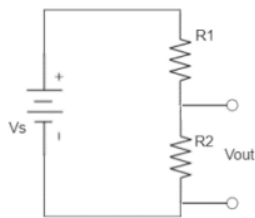
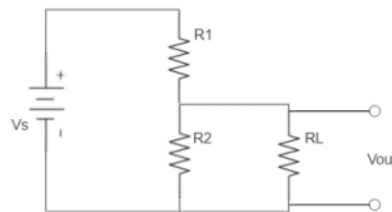
- Returns the **sine** of an angle of **x radians**.
- format

```
#include <cmath>
double sin (double x);
float sin (float x);
long double sin (long double x);
```

ASSIGNMENT 1-3. 2

Assignment 1-3. 2

- Write a program to compute V_{out} provided that the program was given with V_1 , R_1 , R_2 and R_L in the following circuit. The value of V_{out} can be derived by the simplified formula illustrated as below. On the other hand, the total power ratio received by the load is the ratio of the power delivered to load resistor compared to the power supplied by the power supply unit. The final output voltage shall be represented by a form of “**{numerator}/{denominator}**” appended with a form of decimal (or a form of repeating decimal if it is needed). The total power ratio received by the load shall be expressed rounded to the nearest hundredth decimal place.

Figure 2.1 when R_L is set to zeroFigure 2.2 when R_L is set to a non-zero value

$$V_{out} = \frac{V_s(R_2 || R_L)}{(R_1 + R_2 || R_L)}, \text{load_power_ratio} = \frac{P_{Load}}{P_s}$$

$$(R_2 || R_L = \frac{R_2 R_L}{R_2 + R_L}, P_L = V_L I_L, P_s = V_s I_s)$$

Input	Output
Vs: 40 R1: 4 R2: 2 R(Load): 0	Vout: 80/6 = 1(3.)
Vs: 10 R1: 4 R2: 2 R(Load): 2	Vout: 40/20 = 2 Load power ratio: 10.0%

Assignment 1-3. 2

- Calculate value V_{out} in circuit when R_L is set to zero

Ohm's law

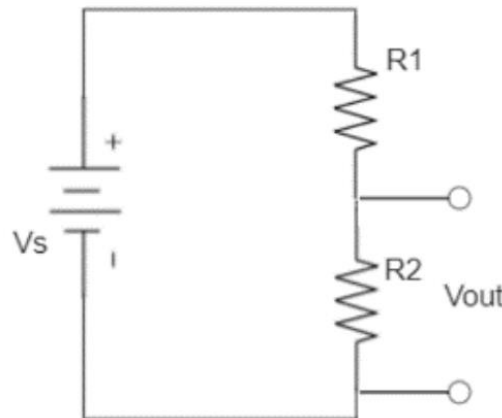
$$V = I \cdot R$$

① 전류 $I = \frac{V_S}{(R_1 + R_2)}$

② 전압 $V_{out} = I \cdot R_2 = \frac{V_S}{(R_1 + R_2)} \cdot R_2$

* 파워 $P_S = V_S \cdot I$

$$P_L = V_L \cdot I$$



Assignment 1-3. 2

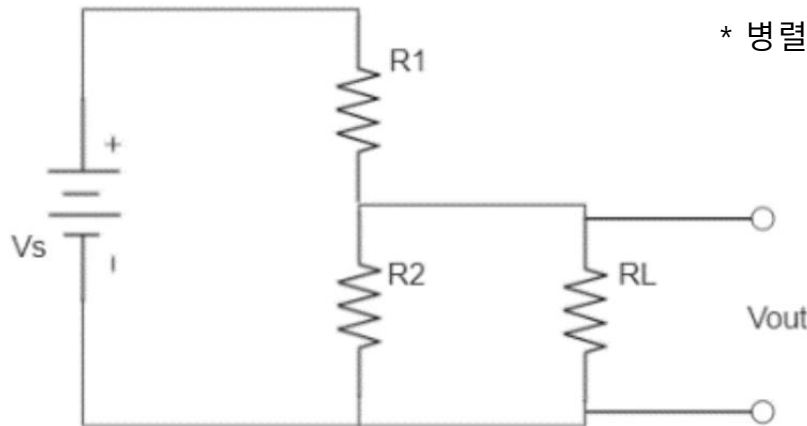
- Calculate value V_{out} in circuit when R_L is set to a non-zero value

Ohm's law

$$V = I \cdot R$$

① 전류 $I = \frac{V_S}{(R_1 + R_2 || R_L)}$ * 병렬 저항 $R_2 || R_L = \frac{R_2 \cdot R_L}{R_2 + R_L}$

② 전압 $V_{out} = I \cdot R_2 = \frac{V_S}{(R_1 + R_2 || R_L)} \cdot (R_2 || R_L)$ * 파워 $P_S = V_S \cdot I$
 $P_L = V_L \cdot I_{RL}$



* 병렬 전류 $I_{RL} = \frac{R_2}{R_2 + R_L} \cdot I$

ASSIGNMENT 1-3. 3


Assignment 1-3. 3

- Write a program that converts a number from a specific base to another desired base (one of **binary**, **octal**, **decimal**, **hexadecimal**).


Input	Output
181 10 2	10110101
10110101 2 8	265
265 8 16	B5

Assignment 1-3. 3


■ 10진법 변환

2	181	...	1	
2	90	...	0	
2	45	...	1	
2	22	...	0	
2	11	...	1	
2	5	...	1	
2	2	...	0	
2	1	...	1	
0				

$$181_{10} \rightarrow 10110101_2$$

8	181	...	5	
8	22	...	6	
8	2	...	2	
0				

$$181_{10} \rightarrow 265_8$$

16	181	...	5	
16	11	...	11	
0				

$$181_{10} \rightarrow B5_{16}$$

Assignment 1-3. 3

■ 2진법 변환

10110101
└─┘ └─┘ └─┘
↓ ↓ ↓
2 6 5

$10110101_2 \rightarrow 265_8$

10110101
└─┘ └─┘
↓ ↓
B 5

$10110101_2 \rightarrow B5_{16}$

Assignment 1-3. 3

- 구현 간 format flag 사용 금지

```
int main()
{
    int x = 181;

    cout.setf(ios::dec, ios::basefield);
    cout << x << endl;

    cout.setf(ios::oct, ios::basefield);
    cout << x << endl;

    cout.setf(ios::hex, ios::basefield);
    cout << x << endl;

    return 0;
}
```

```
181
265
b5
```

ASSIGNMENT 1-3. 4

Assignment 1-3. 4

- Write a program that sorts the input integers in ascending order, extracts the median number, implements four conventional algorithms (**insertion sort**, **quick sort**, **merge sort**, and **bubble sort**) and then **compares their time complexity** to yield the result. It is assumed that **the number of inputs is odd**. Note that **you should give the program the number of inputs in advance**.

Input	Output
15 26 9 23 19 1 35 4 31 14 13 28 3 18 33 19	Sorted order: 1 3 4 9 13 14 18 19 19 23 26 28 31 33 35 Median number: 19

Assignment 1-3. 4

■ Bubble sort

①

3	6	2	5	8	4	1	7
---	---	---	---	---	---	---	---

3	6	2	5	8	4	1	7
---	---	---	---	---	---	---	---

swap

3	2	6	5	8	4	1	7
---	---	---	---	---	---	---	---

swap

3	2	5	6	8	4	1	7
---	---	---	---	---	---	---	---

3	2	5	6	8	4	1	7
---	---	---	---	---	---	---	---

swap

3	2	5	6	4	8	1	7
---	---	---	---	---	---	---	---

swap

3	2	5	6	4	1	8	7
---	---	---	---	---	---	---	---

swap

3	2	5	6	4	1	7	8
---	---	---	---	---	---	---	---

②

3	2	5	6	4	1	7	8
---	---	---	---	---	---	---	---

swap

2	3	5	6	4	1	7	8
---	---	---	---	---	---	---	---

•

•

•

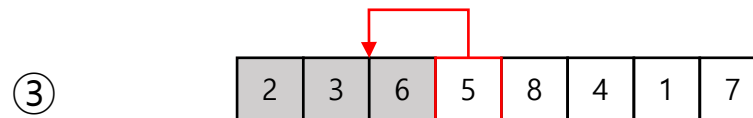
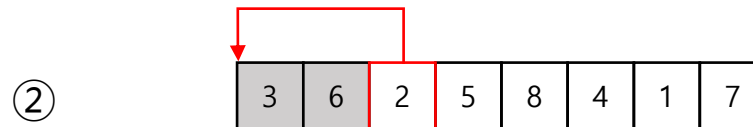
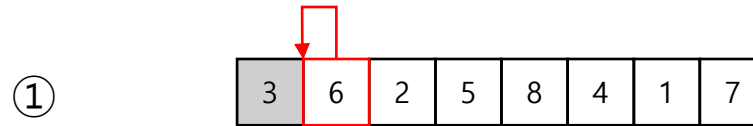
Assignment 1-3. 4

- Bubble sort source code

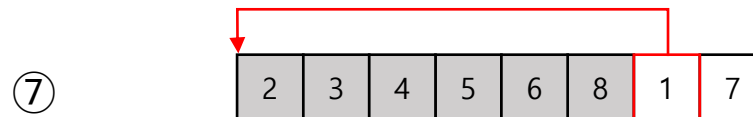
```
void bubbleUp(int list[], int current, int last) {  
    for (int walker = last; walker > current; walker--) {  
        if (list[walker] < list[walker - 1]) {  
            int temp = list[walker];  
            list[walker] = list[walker - 1];  
            list[walker - 1] = temp;  
        }  
    }  
    return;  
}  
  
void bubbleSort(int list[], int last) {  
    for (int current = 0; current < last; current++)  
        bubbleUp(list, current, last);  
    return;  
}
```

Assignment 1-3. 4

■ Insertion sort



⋮



Assignment 1-3. 4

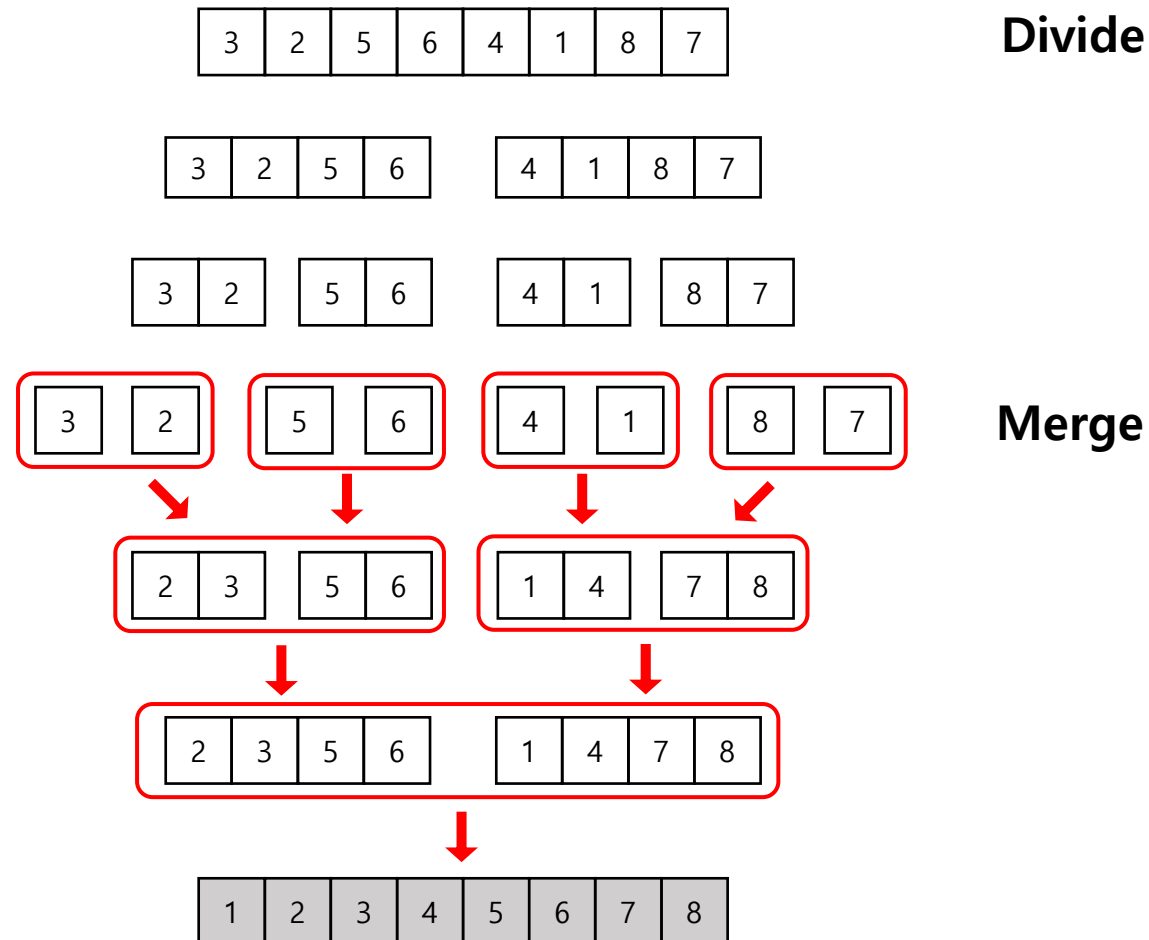
- Insertion sort source code

```
void insertOne(int list[], int current) {
    bool located = false;
    int temp = list[current];
    int walker;
    for (walker = current - 1; walker >= 0 && !located; ) {
        if (temp < list[walker]) {
            list[walker + 1] = list[walker];
            walker--;
        }
        else located = true;
    }
    list[walker + 1] = temp;
    return;
}

void insertionSort(int list[], int last) {
    for (int current = 1; current <= last; current++)
        insertOne(list, current);
    return;
}
```

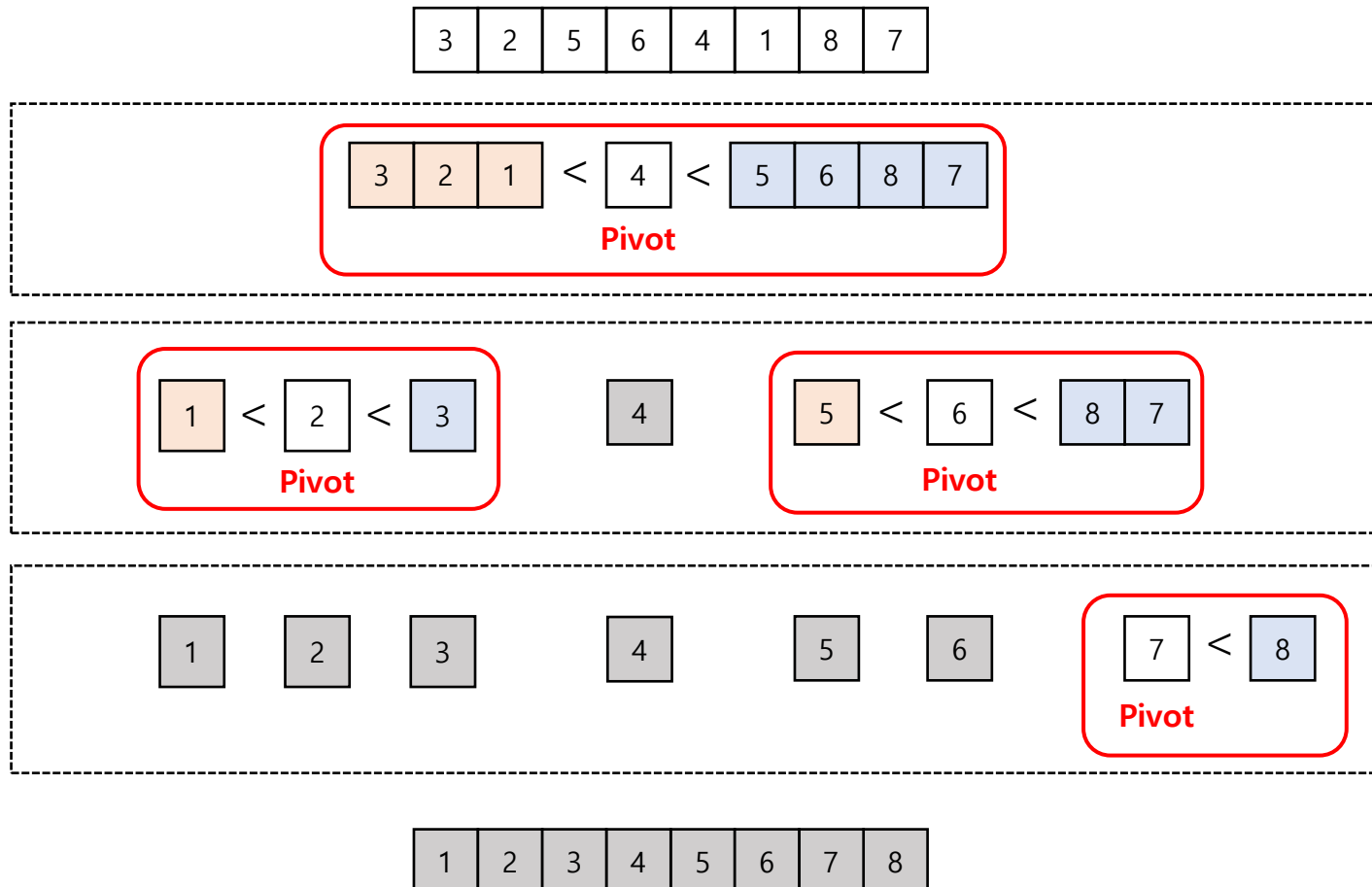
Assignment 1-3. 4

■ Merge sort



Assignment 1-3. 4

▪ Quick sort



과제 제출 방법

과제 제출 방법

▪ FTP Upload (Klas 과제 제출 X)

- Address : <ftp://223.194.8.1:1321>
- username : IPSL_OBJ
- password : ipslobj_2023

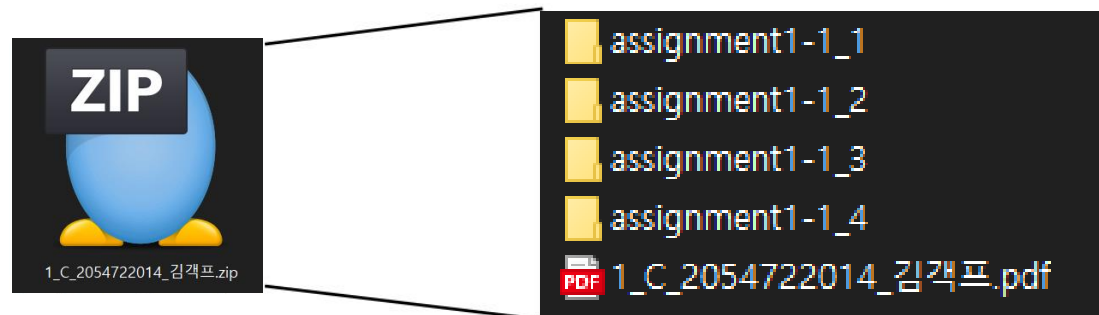
▪ Due date

- Soft copy: 마감일 3/31(금) 23:59:59까지 제출 (서버시간 기준)
- Delay
 - 마감일 이후 +7일까지 제출 가능
 - 단, 1일 초과마다 과제 총점의 10%씩 감점

과제 제출 방법

▪ Soft copy

- 과제(보고서, 소스 코드)를 압축한 파일 제출
 - 설계반_실습반_학번_이름.zip
 - 예) 설계1반 수강, 실습 A반: 1_A_학번_이름.zip
 - 예) 설계 수강, 실습 미수강: 2_0_학번_이름.zip
 - 예) 설계 미 수강, 실습 C반: 0_C_학번_이름.zip



- 과제 수정하여 업로드 시 버전 명시
 - 설계반_실습반_학번_이름_verX.zip

과제 제출 방법

▪ Soft copy

– 과제 보고서

- 영문 또는 한글로 작성
- **반드시 PDF**로 제출 (PDF 외 파일 형식으로 제출시 0점 처리)
- 보고서 양식
 - 문제 및 설명(문제 capture 금지) / 결과 화면 / 고찰
 - 보고서 양식은 아래 경로에서 참고
 - <https://www.ipsl.kw.ac.kr/post/1%EC%B0%A8-%EA%B3%BC%EC%A0%9C>
- 소스코드 제외
- 분량 제한 없음
- **표절 적발 시 0점 처리**

– 소스 코드

- Visual Studio 2022 community 사용 필수
 - <https://docs.microsoft.com/ko-kr/visualstudio/install/install-visual-studio?view=vs-2022>
- STL (Standard Template Library) 사용 금지 (vector, map, algorithm 등)
- Debug 폴더를 제외한 모든 파일 제출
 - .sln 파일 포함(.cpp 만 제출하지 말것)
- **각 문제마다 프로젝트 파일 생성 필수**
- **주석 반드시 달기**
- **소스코드 표절 적발 시 0점 처리**

END OF PRESENTATION

Q&A