4190.103A-001: Programming Practice Lab. 9

Jaewon Choi
Keunsan Park
Haeun Lee
(snucsl.ta@gmail.com)

Systems Software & Architecture Lab.

Seoul National University

Spring 2023

Lab. 8(1) 실습 풀이

실습 1

Q. 입력받은 정수에서 0 ~ 9 의 갯수를 세고, 이를 내림차순으로 출력해보자.

첫째줄에 정수 N을 입력받는다.
(1 ≤ N ≤ 30,000,000)
두번째줄에 N개의 정수를 입력받고, 0 ~ 9가 각 몇 번 입력됐는지 세고, 내림차순으로 출력한다.

조건

- 입력된 횟수가 0번이면 출력하지 않는다.
- 입력받는 정수는 0 ~ 9 사이의 정수이다.

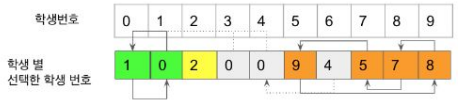
```c
int main(){
    /* Write your code here */
    int n;
    int cnt[10] = { 0, };

    scanf("%d", &n);

    for(int i = 0; i < n; i++)
    {
        int tmp;
        scanf("%d", &tmp);

        cnt[tmp]++;
    }

    sort(cnt);

    for(int i = 0; i < 10; i++)
    {
        if(cnt[i] != 0)
            printf("%d ", cnt[i]);
    }

    return 0;
}
```

```c
void swap(int *a, int *b)
{
    int tmp = *a;
    *a = *b;
    *b = tmp;
}

void sort(int *cnt){
    /* Write your code here */
    for (int i = 0; i < 9; i++)
    {
        for (int j = 9; j > i; j--)
        {
            if (cnt[j-1] < cnt[j])
                swap(&(cnt[j]), &(cnt[j-1]));
        }
    }
}
```

실습 2

Q. 입력받은 정수에 따라 구성된 팀의 총 수와 각 팀의 구성을 출력해보자.

10명의 학생이 있으며 각 학생의 번호는 0 ~ 9 이다.
0번부터 순서대로 자신이 함께하고 싶은 학생의 번호를 선택할 때(자기 자신 포함), 아래 조건에 따라서 **형성되는 팀의 총 수**와 그에 따른 **각 팀의 구성원**을 출력해라.



조건

- 0부터 9 사이의 정수 10개를 입력받는다.
- S1 -> S2, S2 -> S3, ... Sn -> S1 과 같이 구성되어야 한다.
- 예를 들어, 위 그림에서 3번 학생과 4번 학생은 0번을 선택했지만, 선택한 번호를 따라갔을 때 자기 자신의 번호가 없으므로 팀이 되지 않는다.
- 만약 위 그림에서 1번 학생이 3번, 3번 학생이 4번 학생을 선택했다면, `0, 1, 3, 4` 번 학생들은 한 팀이 된다.

```c
/* Step 2: 팀을 이룰 수 있다면,
   같은 팀으로 묶인 사람들에게 팀번호를 부여하여 팀 구성*/
start = student_id;
next = start;

while(1)
{
    if(visited[next] == 1)
        break;

    visited[next] = 1;
    next = choice[next];
}

if(next == start)
{
    (*team_cnt)++;

    for(int i = 0; i < ARRAY_SIZE; i++)
    {
        if(visited[i] == 1)
            team[i] = *team_cnt;
    }
}
```

실습 3

Q. 주어진 문자열을 사용해 규칙대로 출력해보자.

입력의 첫번째 줄에 N(1 ≤ N ≤ 100)과 임의의 문자 가 띄어쓰기로 구분되어 입력된다.
이 때, 입출력 예제를 보고 규칙을 유추하고, 입력된 문자열을 활용하여 출력해보자.

입력 1

```
5 *
```

출력 1

```
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

```
/* Step 1 )
* n 값에 따라 만들어야 할 2차원 배열의 width, height를 생각해보고
* 동적할당을 통해 result 배열에 할당한 뒤, 알맞은 문자를 넣습니다.
* 7주차 과제에서 2차원 동적할당을 어떻게 받았었는지 떠올려 봅시다.
*/
scanf("%d %c",&n,&c);
result = (char**)malloc(sizeof(char*)*(2*n-1));
for(int i=0;i<2*n-1;i++)
    result[i] = (char*)malloc(sizeof(char)*(2*n-1));

for(i=0;i<n;i++){
    for(j = 0;j < n-i-1;j++)
        result[i][j] = ' ';
    for(j;j < 2*n-1-(n-i-1);j++)
        result[i][j] = c;
    for(j;j < 2*n-1;j++)
        result[i][j] = ' ';
}

for(i;i<2*n-1;i++){
    for(j=0;j < i-n+1;j++)
        result[i][j] = ' ';
    for(j;j < 2*n-1-(i-n+1);j++)
        result[i][j] = c;
    for(j;j < 2*n-1;j++)
        result[i][j] = ' ';
}

printDiamond(2*n-1, 2*n-1, result);
return 0;
```

```
/* Step 2 )
* 2차원 배열의 width, height를 활용해 배열을 출력합니다.
*/
for(int i=0;i<height;i++){
    for(int j=0;j<width;j++){
        printf("%c",arr[i][j]);
    }
    printf("\n");
}
```

실습 4

Q. 수확 가능한 최대 오렌지 갯수를 세어보자.

농장에 k개의 오렌지 나무가 있다. k 일동안 하루에 하나의 나무만 정해서 해당 나무에서 자란 오렌지를 모두 수확한다.
농장에는 비법 비료가 있어, 해당 비료를 뿌리고 밤이 지나면 오렌지가 매우 빨리 열리고, 열리는 갯수는 나무마다 다르다.
어느 나무에서 먼저 수확하느냐에 따라서 총 구할 수 있는 오렌지 갯수는 달라지는데, 이 때 최대 구할수 있는 오렌지 갯수를 출력해라.

조건

- 같은 나무에서 여러번 수확할 수 없다. (하나의 나무에서 한 번만 수확 가능)
- 첫째 줄에 k 가 입력된다.
- 둘째 줄에 처음 각 오렌지 나무에 있었던 오렌지 갯수들이 입력된다.
- 셋째 줄에 각 오렌지 나무마다 하룻동안 새로 열리는 오렌지 갯수가 입력된다.

```c
int main(){
    /* TIP)
    * 정렬 알고리즘을 활용해 최대값을 어떻게 구할지 생각해봅니다.
    */
    int i, k, sum=0;
    int *orange, *new;

    scanf("%d", &k);

    orange = (int *)malloc(sizeof(int)*k);
    new = (int *)malloc(sizeof(int)*k);

    for(int i = 0; i < k; i++)
        scanf("%d", &(orange[i]));

    for(int i = 0; i < k; i++)
        scanf("%d", &(new[i]));

    sort(new, k);

    for(int i = 0; i < k; i++)
    {
        sum += orange[i];
        sum += (new[i] * i);
    }

    printf("%d", sum);

    return 0;
}
```

Lab. 8(2) 실습 풀이

실습 1

친구 바꿔서 마니또 진행

문제의 입력으로 N을 입력받는다. N은 학생 수를 나타내며, 학생들은 0~(N-1) 사이의 학생 번호를 가진다. 각 학생은 선물을 주어야 할 친구가 있고, 이에 대한 정보는 아래와 같이 입력으로 주어진다.

선물을 주어야 할 친구의 수	친구들의 학생 번호		
1	1		
3	0	2	3
2	1	3	
1	1		

각 줄 i 에는 학생 번호 i를 가진 학생이 선물을 주어야 할 총 친구의 수와, 친구들의 학생 번호가 차례대로 입력된다. 총 N명이 있으므로 총 N 줄이 입력으로 주어진다.

이 때, 선물 교환을 재미있게 하기 위해 선생님은 새로운 제안을 하였다. M 쌍의 학생들이 선물을 주어야 할 친구를 교환하자는 것이다. 교환 이후 모든 학생들이 선물을 주어야할 학생들의 학생 번호를 각각 한 줄에 출력하시오. 만약 바뀜에 따라 본인에게 선물을 주어야 한다면 본인에게 선물을 준다고 가정한다.

```c
for (int i = 0; i < N; i++) {
    scanf("%d", &FRIENDS[i][0]);
    for (int j = 0; j < FRIENDS[i][0]; j++) {
        // Step 1 : 각 학생에 대해 선물을 주어야 할 친구들을 입력 받을 것
        // Write your code here
        scanf("%d", &FRIENDS[i][j + 1]);
    }
}
for (int i = 0; i < M; i++) {
    int n1, n2;
    int * tmp;
    scanf("%d %d", &n1, &n2);
    // Step 2 : 선물을 주어야 할 친구들을 가리키는 포인터를 맞 교환할 것
    // Write your code here
    tmp = FRIENDS[n1];
    FRIENDS[n1] = FRIENDS[n2];
    FRIENDS[n2] = tmp;
}
// Step 3 : 바뀐 선물을 주어야 할 친구들의 리스트를 각 줄에 출력할 것
// Write your code here
for (int i = 0; i < N; i++) {
    for (int j = 0; j < FRIENDS[i][0]; j++) {
        printf("%d ", FRIENDS[i][j + 1]);
    }
    printf("\n");
}
```

실습 2

가계 조사

사람 수 N을 입력으로 받자. 각 사람의 사람 번호는 0~N-1 사이에 있다. 이후 입력으로 각 사람의 부모님의 번호가 입력이 된다. 각 사람마다 자손의 총 수가 몇 명인지 출력하시오. 가장 조상의 경우 부모님의 번호는 -1로 입력된다.

```c
void Update_Num_Descendents(int * PAR, int * NUM_DESCENTS, int * N) {
    int changed = 0;
    // Step 1 : 함수 수정
    // Write your code here
    for (int i = 0; i < *N; i++) {
        if (PAR[i] != -1 && Fully_Updated(PAR, N, i)) {
            changed = 1;
            NUM_DESCENTS[PAR[i]] += NUM_DESCENTS[i] + 1;
            PAR[i] = -1;
        }
    }

    if (changed)
        Update_Num_Descendents(PAR, NUM_DESCENTS, N);
    else
        return;
}
```

```c
// Step 2 : Update_Num_Descendents 함수를 사용하여 NUM_DESCENDENTS 배열을 올바르게 초기화
Update_Num_Descendents(PAR, NUM_DESCENTS, N);
// Step 3 : 자손의 수 출력
for (int i = 0; i < (*N); i++) {
    printf("%d ", NUM_DESCENTS[i]);
}
```

실습 3

포인터 배열을 이용하여 순서 바꾸기

1부터 10까지 숫자가 적힌 const int 배열 ARR이 있다. 해당 배열은 const 배열이기 때문에 배열의 내용을 바꿀 수 없다. int pointer 배열인 ADDR을 선언하여 ARR 배열 각 원소의 주소를 저장한다. 이후, N을 정수 입력으로 받고, N개 만큼의 0~9 사이의 두 숫자쌍을 입력으로 받아 포인터 배열의 값을 swap 한다. 이후, 포인터 배열을 사용하여 바뀐 순서를 출력하시오.

입력 1

```
1
1 2
```

출력 1

```
1 3 2 4 5 6 7 8 9 10
```

```c
// Step 1 : ADDR 배열 초기화
for (int i = 0; i < 10; i++) {
    ADDR[i] = &ARR[i];
}
int t1, t2;
for (int i = 0; i < N; i++) {
    // Step 2 : 두 숫자를 입력받고 ADDR 배열의 값을 맞 교환
    scanf("%d %d", &t1, &t2);
    int * tmp = ADDR[t1];
    ADDR[t1] = ADDR[t2];
    ADDR[t2] = tmp;
}
// Step 3 : 바뀐 배열의 내용을 출력
for (int i = 0; i < 10; i++) {
    printf("%d ", *ADDR[i]);
}
```

Lab. 8 과제 풀이

과제 1

Q. S의 최솟값을 구해보자

길이가 N인 정수 배열 A와 B가 있을 때
함수 S는 다음과 같이 정의된다.

$$S = A[0] \times B[0] + \ldots + A[N-1] \times B[N-1]$$

S의 값을 가장 작게 만들기 위해서 A 배열의 수를
재배열하자. 단, B 배열의 수는 재배열하면 안된다.

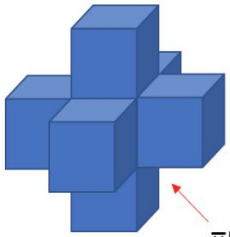
입력

- 첫째 줄에 N이 주어진다.
- 둘째 줄에는 배열 A의 원소가 순서대로 N개 주어진다.
- 셋째 줄에는 배열 B의 원소가 순서대로 N개 주어진다.
- N은 50보다 작거나 같고, A와 B의 각 원소는 100보다 작거나 같은 음이 아닌 정수이다.

```
scanf("%d",&N);

A = (int*)malloc(sizeof(int)*N);
B = (int*)malloc(sizeof(int)*N);
B_sorted_idx = (int*)malloc(sizeof(int)*N);

for(i=0;i<N;i++)
    scanf("%d",A+i);
for(i=0;i<N;i++)
    scanf("%d",B+i);

sort(A, N);

for(i=100; i>=0; i--){
    for(j=0; j<N; j++){
        if(i == B[j]){
            B_sorted_idx[cur] = j;
            cur++;
        }
    }
}

for(i=0;i<N;i++)
    sum += B[B_sorted_idx[i]] * A[i];

printf("%d\n",sum);
```

과제 2

3차원 공간 상에 1 X 1 X 1 크기의 정육면체가 있다. 이 때 정육면체의 각 면의 방향으로 정육면체를 쌓아 아래와 같은 도형을 만들 수 있다.



팔의 길이 L = 1

N이라는 숫자를 입력받자. 해당 숫자를 이용하여 3차원 공간을 나타내는 3차원 배열을 할당 받고, 총 N X N X N개의 숫자를 입력받아 해당 배열을 채울 것이다. 이렇게 배열을 채우는 과정은 코드에서 제공이 되며, 가장 위 층부터 아래 층까지 입력이 주어진다. 그림에서 정육 면체 하나가 배열의 원소 1개이다. 배열에 1이 있다면, 해당 공간은 도형이 있을수 없는 공간이며, 0이 있다면 해당 공간에 도형이 있을 수 있다. 주어진 공간에서 존재할 수 있는 도형의 최대 팔의 길이를 출력하시오.

```
int *** SPACE = (int ***)malloc(sizeof(int **) * N);
// Step 1 : 3차원 공간 할당을 마무리 하시오, malloc 사용.
for (int i = 0; i < N; i++) {
    SPACE[i] = (int **) malloc(sizeof(int *) * N);
    for (int j = 0; j < N; j++) {
        SPACE[i][j] = (int *)malloc(sizeof(int) * N);
    }
}
```

```
// Step 2 : solve 함수를 완성하시오.
// 제공된 IN_BOUNDARY 함수를 사용할 수 있음.
int solve(int *** SPACE, int i, int j, int k) {
    int l = 0;
    while (IN_BOUNDARY(i - l, j, k) && IN_BOUNDARY(i + l, j, k)
            && IN_BOUNDARY(i, j - l, k) && IN_BOUNDARY(i, j + l, k)
            && IN_BOUNDARY(i, j, k - l) && IN_BOUNDARY(i, j , k + l)
            && !SPACE[i - l][j][k] && !SPACE[i + l][j][k]
            && !SPACE[i][j - l][k] && !SPACE[i][j + l][k]
            && !SPACE[i][j][k - l] && !SPACE[i][j][k + l]
            ) {
        l++;
    }
    if (l > 0) l--;
    return l;
}
```

Stack & Queue

Stack

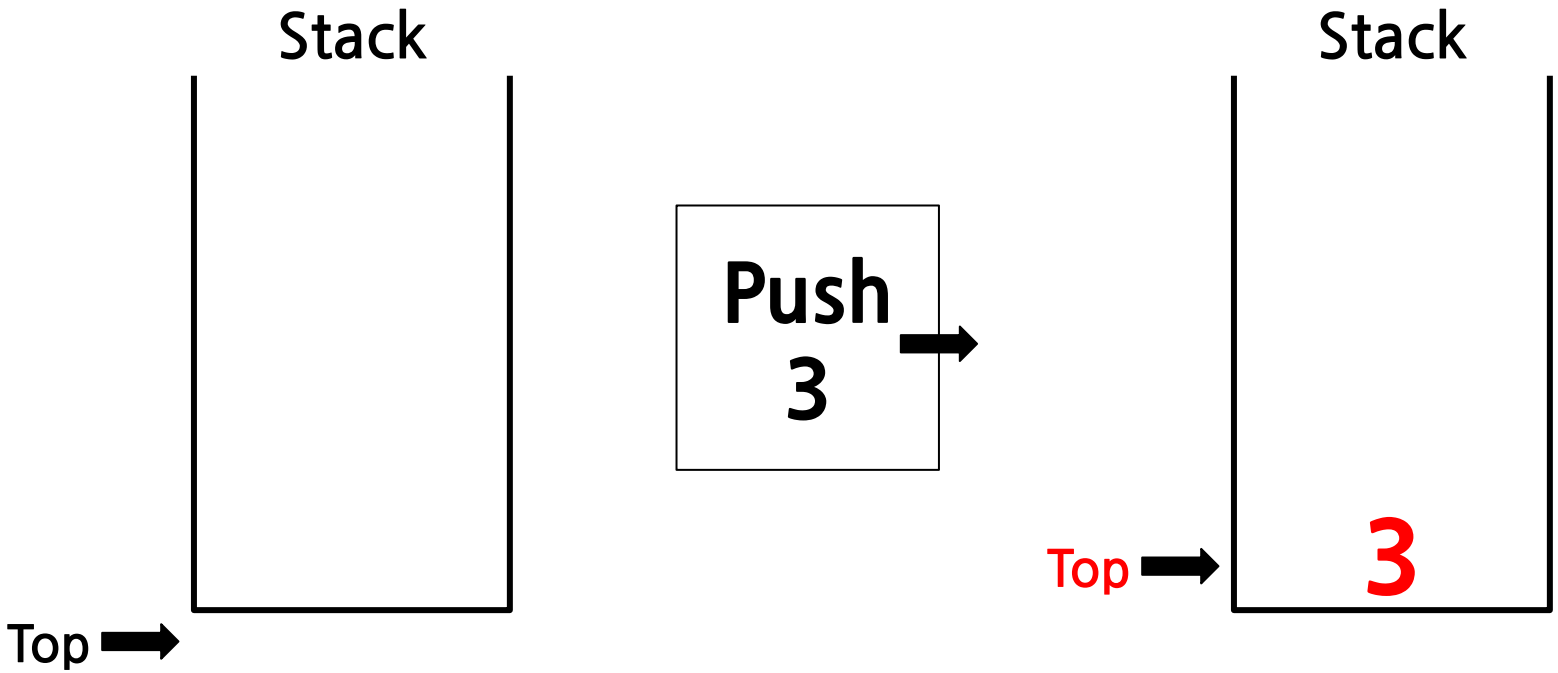
- Stack과 Queue에 대해서 배워봅시다
- Stack은 LIFO(Last In First Out) 구조를 갖고 있습니다
- Queue는 FIFO(First In First Out) 구조를 갖고 있습니다

Stack

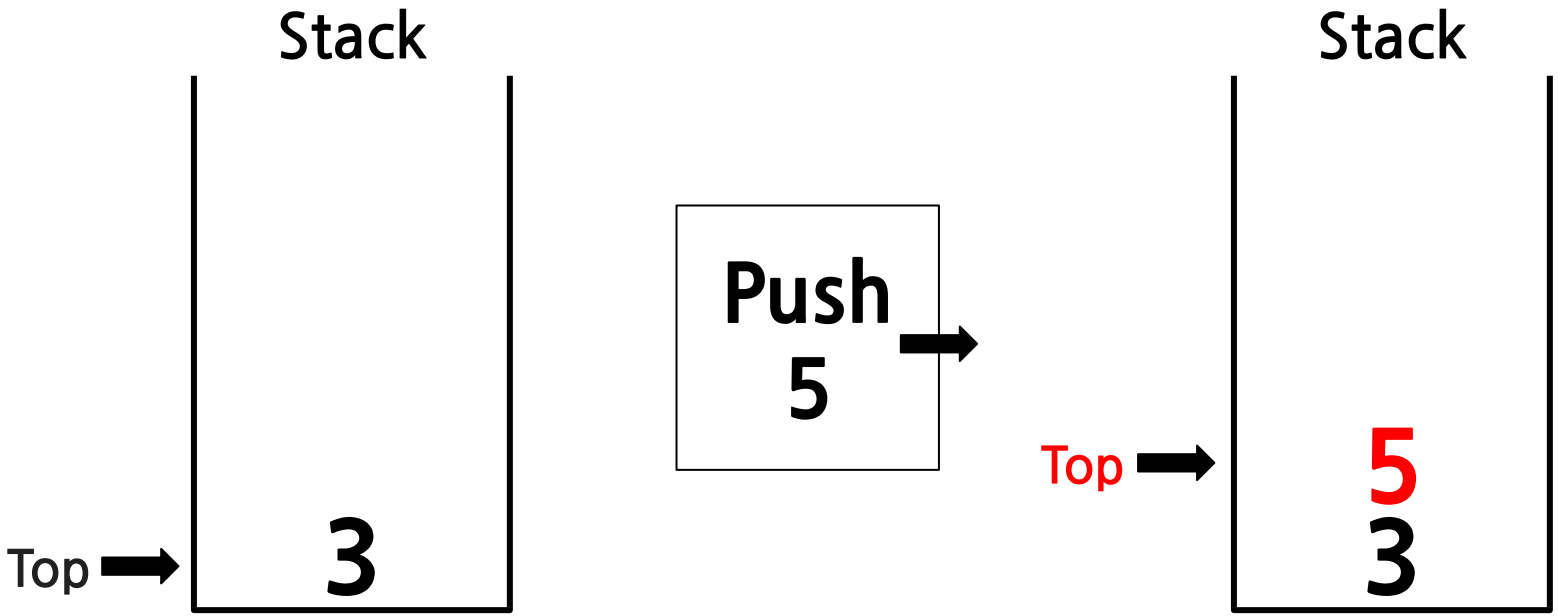
Stack

- Stack에는 다음과 같은 주요 동작이 있습니다
  - Push : Data를 Stack의 Top에 저장
  - Pop : Stack의 Top에 있는 Data를 꺼냄

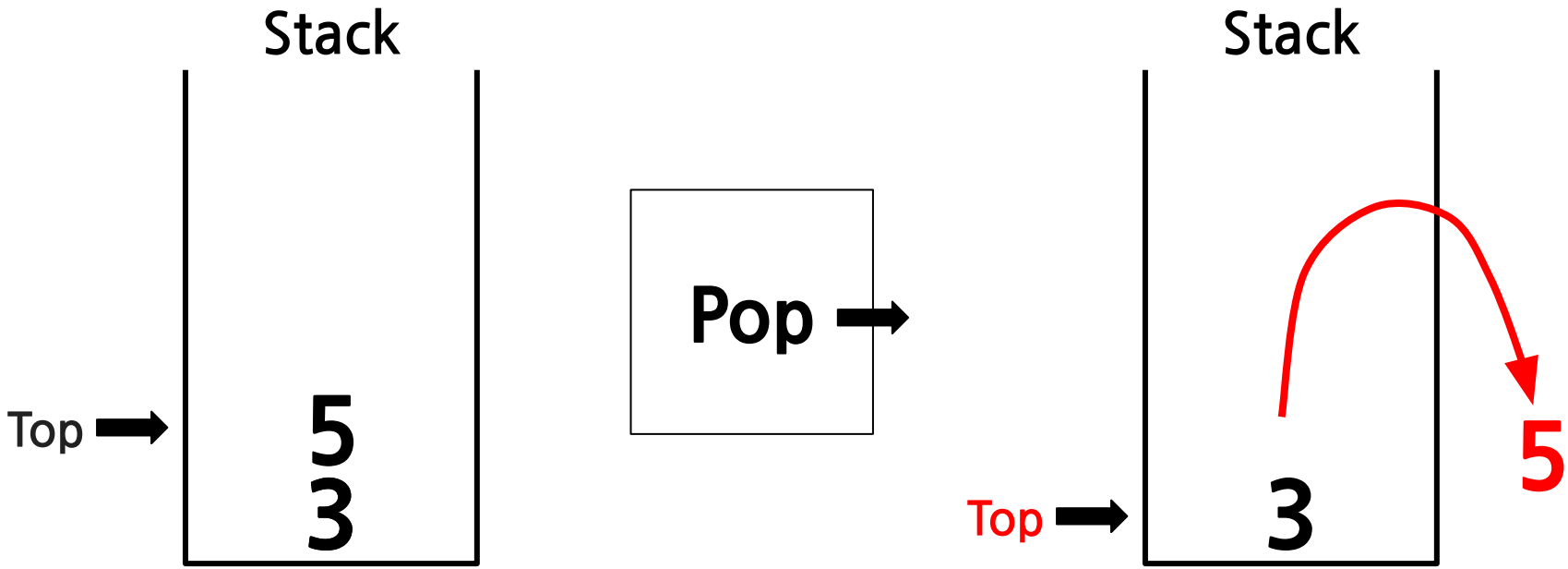
Stack - Push



Stack - Push



Stack - Pop



Stack - 구현

- 배열을 통해 stack을 구현할 수 있습니다.
  - Stack으로 사용할 배열과 top을 나타낼 변수를 선언합니다
  - top은 -1로 초기화하여 비어있음을 나타냅니다
    - top을 0으로 초기화하여 사용할 수도 있습니다

```c
#include <stdio.h>

#define MAX_NUM 100

int stack[MAX_NUM];
int top = -1;
```

Stack - 구현

- Push 함수
  - top이 Stack의 꼭대기를 가리키고 있는지 확인하여, 가득 차 있다면 -1을 return 합니다.
  - 가득 차 있지 않다면, top을 증가시키고 데이터를 저장합니다.
  - '++'의 위치를 확인해보세요

```
14 ▼ int push(int num){
15     if(top == MAX_NUM-1)
16       return -1;
17     stack[++top] = num;
18     return top;
19   }
```

Stack - 구현

- Pop 함수
  - top이 음수인지 확인하여 stack이 비어있다면 -1(top)을 return 합니다
  - 비어있지 않다면, top에 있는 값을 return하고, top을 줄여줍니다
  - '--'의 위치를 확인해보세요

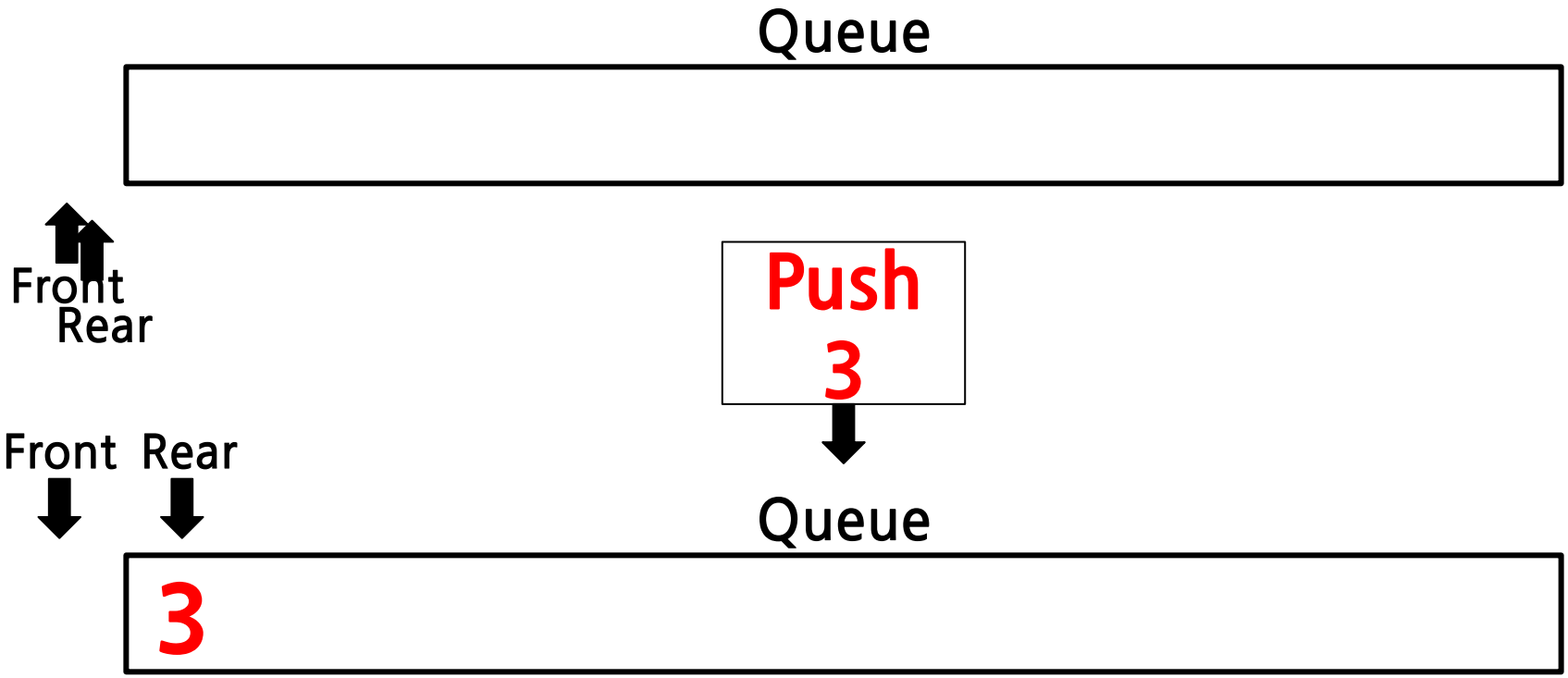
```
 8 ▼ int pop( ){
 9       if(top < 0)
10           return top;
11       return stack[top--];
12   }
```

Queue

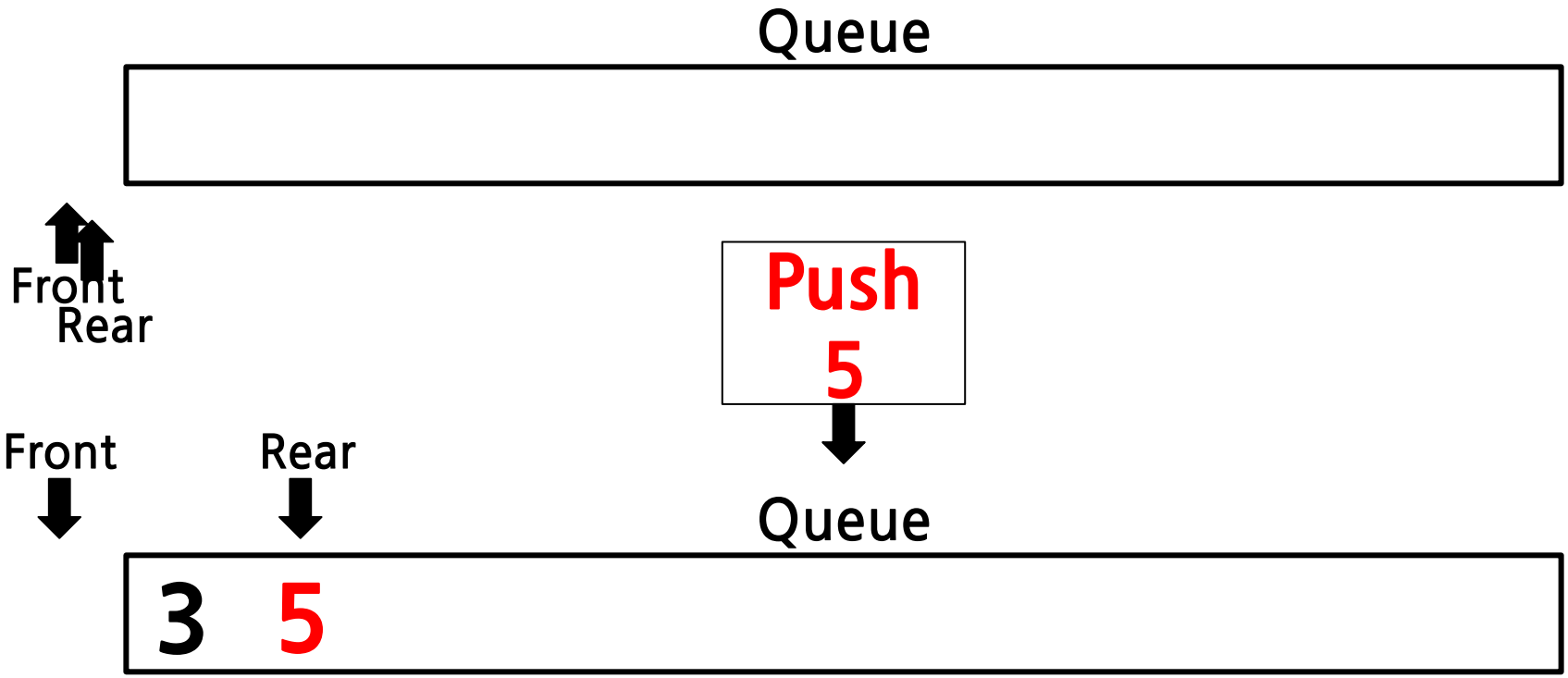
Queue

- Queue에도 Stack과 비슷한 주요 동작이 있습니다
  - Push : Data를 Queue의 rear에 저장
  - Pop : Queue의 front에 있는 Data를 꺼냄

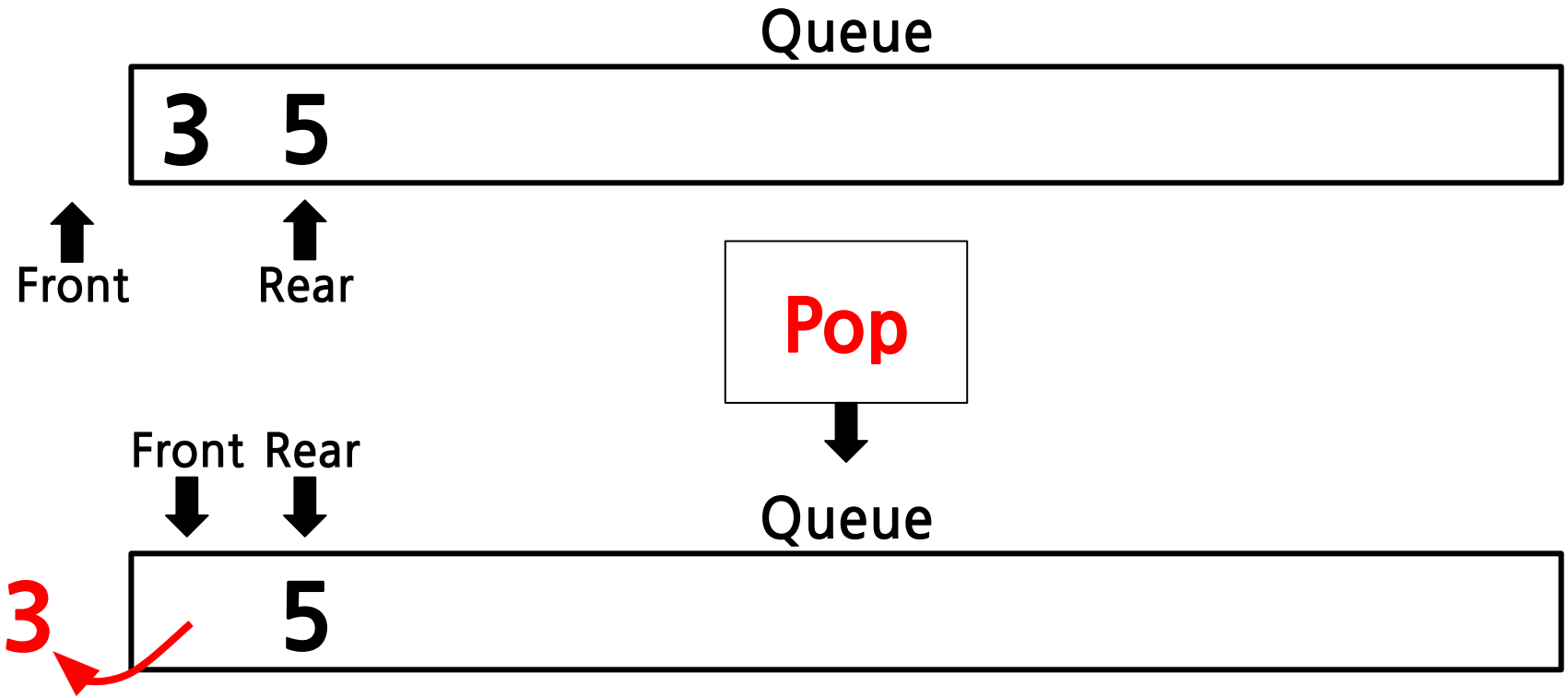
Queue - Push



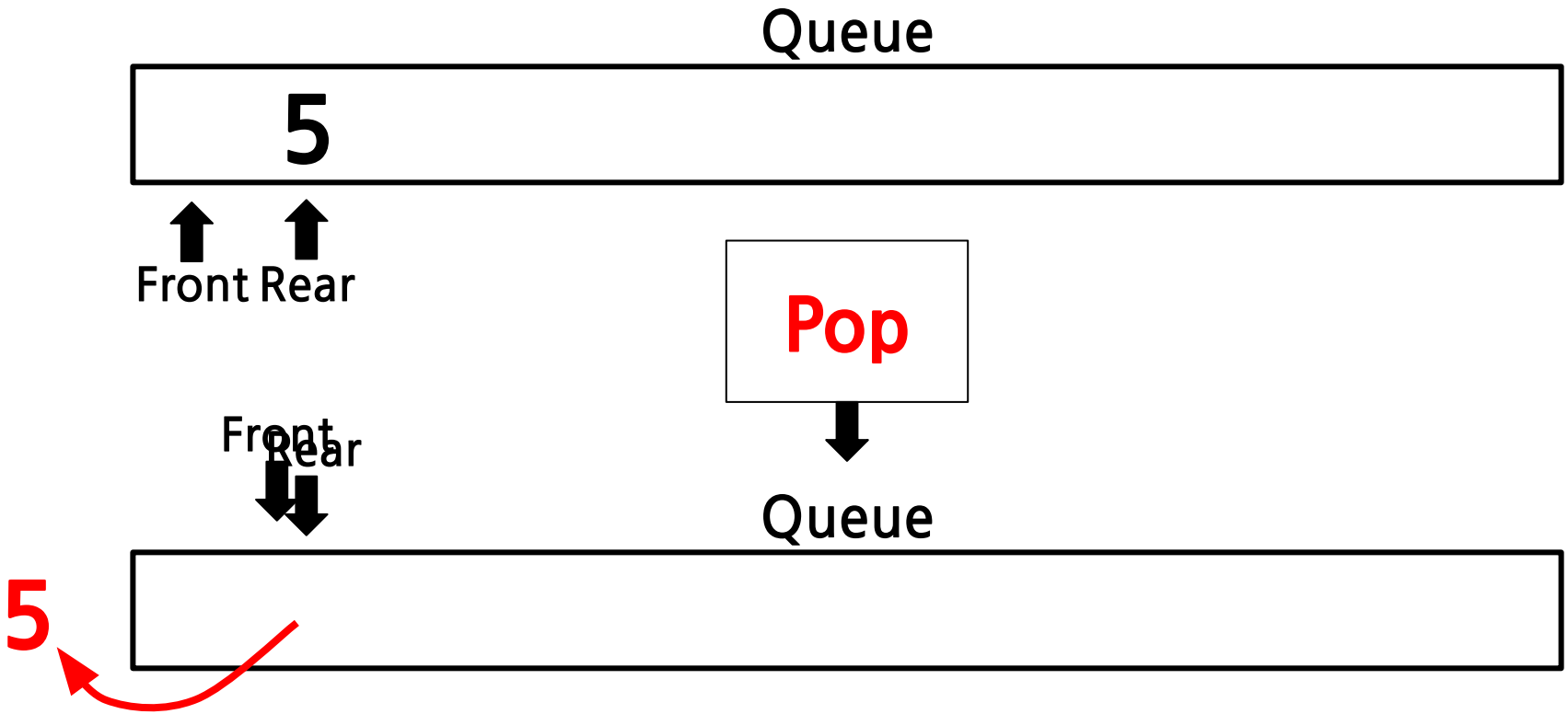
Queue - Push



Queue - Pop



Queue - Pop



Queue - 구현

- 배열을 통해 Queue를 구현 할 수 있습니다.
  - Queue로 사용할 배열과 front와 rear를 나타낼 변수를 선언합니다
  - front와 rear는 -1로 초기화합니다

```c
#include <stdio.h>

#define MAX_NUM 100

int queue[MAX_NUM];
int front = -1, rear = -1;
```

Queue - 구현

- Push 함수
  - rear가 queue에 끝에 도달했는지 확인하여, 도달했다면 -1을 return 하여 오류를 알립니다
  - 아니라면, rear를 증가시키고 data를 저장합니다
  - 그 후, rear를 return 합니다

```
 8 ▼ int push(int num){
 9       if(rear == MAX_NUM-1)
10           return -1;
11       queue[++rear] = num;
12       return rear;
13   }
```

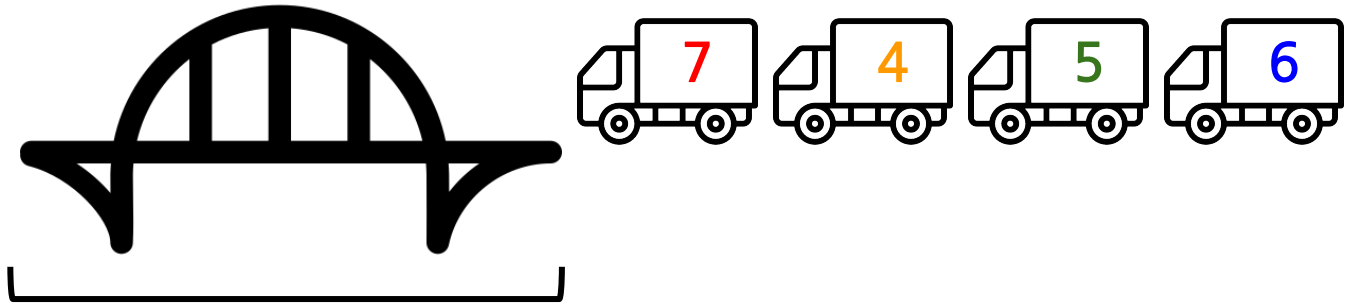
Queue - 구현

- Pop 함수
  - front와 rear를 비교하여 같다면, queue가 비어있다는 뜻이 됩니다
    - -1을 return 하여 오류를 알립니다
  - Queue가 비어있지 않다면, front를 증가시키고 data를 반환합니다

```
15 ▼ int pop(){
16       if(front == rear)
17           return -1;
18       return queue[++front];
19   }
```

실습 & 과제

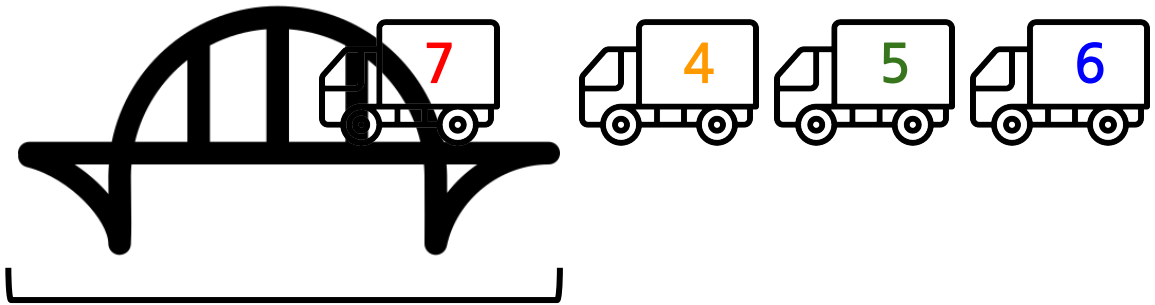
실습 4 - 설명



길이 : 2
최대 무게 : 10

실습 4 - 설명

시간 : 1



길이 : 2
최대 무게 : 10

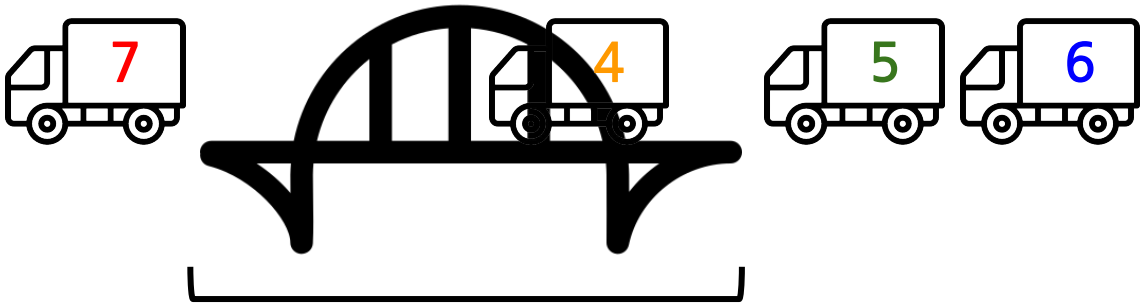
실습 4 - 설명

시간 : 2

길이 : 2
최대 무게 : 10

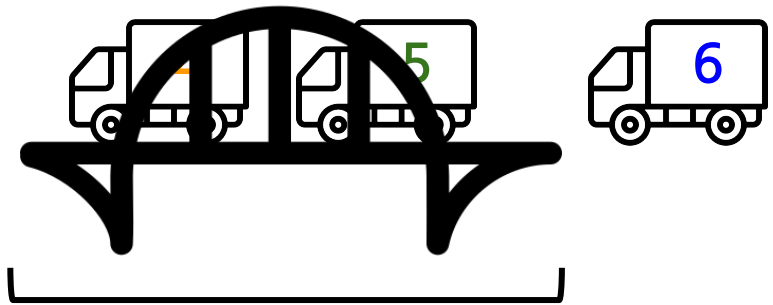
실습 4 - 설명

시간 : 3



실습 4 - 설명

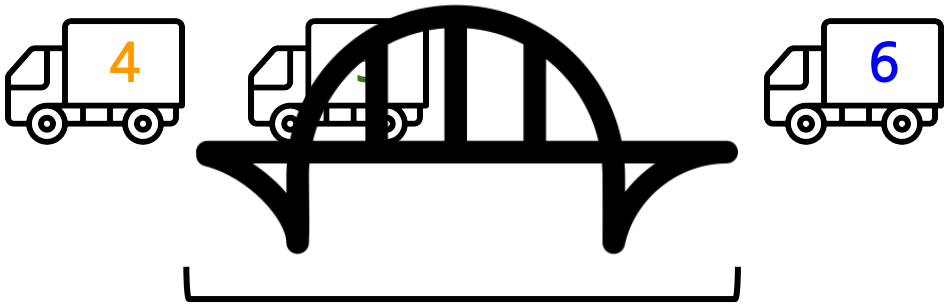
시간 : 4



길이 : 2
최대 무게 : 10

실습 4 - 설명

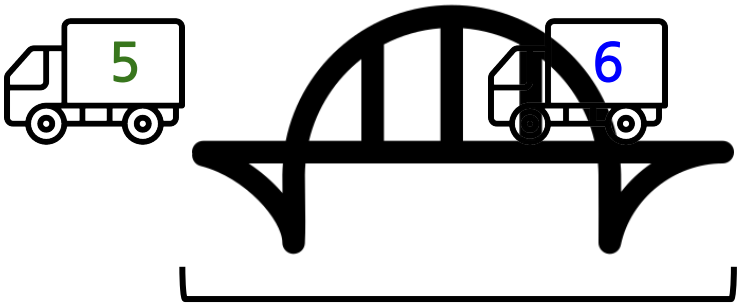
시간 : 5



길이 : 2
최대 무게 : 10

실습 4 - 설명

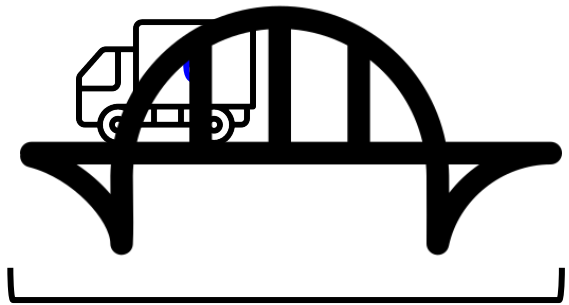
시간 : 6



길이 : 2
최대 무게 : 10

실습 4 - 설명

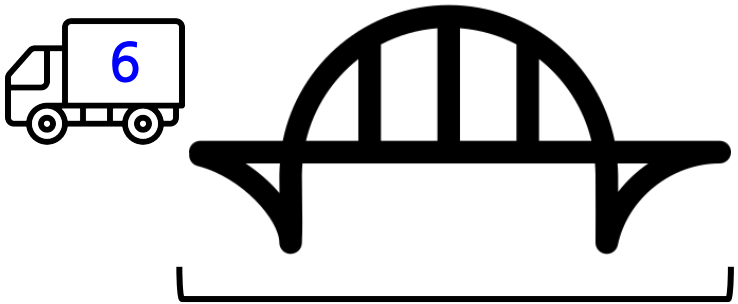
시간 : 7



길이 : 2
최대 무게 : 10

실습 4 - 설명

시간 : 8



길이 : 2
최대 무게 : 10

실습 / 과제

- 과제 내용
  - 당일 진행한 실습과 관련된 문제
- 제출 기한
  - 실습 : 당일 자정까지, 다음날 자정까지 지각제출 허용 (단, 점수 -30%)
  - 과제 : 5/4(금) 23:59 까지, 다음날 자정까지 지각제출 허용 (단, 점수 -30%)
- 제출 방법
  - Elice의 Submit 기능 활용