서울시립대학교 소모임 알

컴퓨터과학부 김희중

1주차 해설

별찍기 - 4

2441

https://www.acmicpc.net/problem/2441

- 행 번호 i가 0번 ~ N-1번으로 진행된다고 할 때
- 공백을 i번, *을 N-i번 출력하는 문제
- 2중 포문을 이용해 아주 간단하게 구현할 수 있다.

2441

https://www.acmicpc.net/problem/2441

- C++ Code(github) : https://github.com/OfficialDominyellow/AlgorithmByDominyellow/blob/master/BackjoonOnlineJudge/2441.cc

X보다 작은 수

10871

https://www.acmicpc.net/problem/10871

- N, X를 입력 받고 시작한다.
- N개의 숫자를 반복문을 통해 입력받는다.
- 입력 받을 때 마다 X와 비교하여 X보다 작을 때만 입력받은 수를 공백과 함께 출력한다.
- printf(“%d “, ...)

10871

https://www.acmicpc.net/problem/10871

- C++ Code(github) : https://github.com/OfficialDominyellow/AlgorithmByDominyellow/blob/master/BackjoonOnlineJudge/10871.cc

큐

10845

https://www.acmicpc.net/problem/10845

- STL에는 자료구조 Queue가 이미 구현되어있다.
- #include <queue> 헤더에 구현되어있다.
- std라는 namespace에 있다. (using namespace std;를 써주어야 큐 선언이 용이하다.)
- queue<자료형> 변수명; 의 형태로 큐를 선언한다.
  (queue<int> q; 라고 선언하면 int형 데이터를 담는 큐가 선언)

10845

https://www.acmicpc.net/problem/10845

- STL의 큐에는 다음과 같은 함수가 구현되어 있다.
- push(데이터); 선언된 자료형에 맞는 데이터를 큐의 back에 삽입한다. Ex)queue<int> q; q.push(3);
- pop(); 큐의 front에서 1개의 원소를 빼낸다.
- front();, back(); 큐의 front, back에 있는 원소를 참조한다. Ex) printf("%d %d\n", q.front(), q.back());

10845

https://www.acmicpc.net/problem/10845

- empty(); 큐가 비어있는지 여부를 bool형 변수(true, false)를 반환하여 알 수 있다. Ex) if(q.empty()) printf(“비어있음”);
- size(); 현재 큐에 몇 개의 원소가 들어있는지 알 수 있다. Ex) printf(“size : %d”, q.size());
- 이 정도의 함수를 사용할 수 있다면 큐 관련 문제를 푸는 과정에서 막힘이 별로 없다.

10845

https://www.acmicpc.net/problem/10845

- 여섯 가지의 명령 중에서 push라는 명령이 들어왔을 때만 int형 변수의 입력을 한번 더 받는다. 나머지 5개의 명령은 주어진 조건에 따라서 출력만 반복적으로 해주면 된다.

- 앞에 설명된 큐 관련 함수를 이용하여 문제를 푼다. empty() 함수의 사용법을 충분히 익히도록 한다.

10845

https://www.acmicpc.net/problem/10845

- C++ Code(github) : https://github.com/OfficialDominyellow/AlgorithmByDominyellow/blob/master/BackjoonOnlineJudge/10845.cc

스택

10828

https://www.acmicpc.net/problem/10828

- STL에는 자료구조 Stack이 이미 구현되어있다.
- #include <stack> 헤더에 구현되어있다.
- std라는 namespace에 있다. (using namespace std;를 써주어야 스택 선언이 용이하다.)
- stack<자료형> 변수명; 의 형태로 큐를 선언한다.
  (stack<int> st; 라고 선언하면 int형 데이터를 담는 스택 선언)

10828

https://www.acmicpc.net/problem/10828

- STL의 스택에는 다음과 같은 함수가 구현되어 있다.
- push(데이터); 선언된 자료형에 맞는 데이터를 큐의 top에 삽입한다. Ex)stack<int> st; st.push(3);
- pop(); 스택의 top에서 1개의 원소를 빼낸다.
- top(); 스택의 top에 있는 원소를 참조한다. Ex) printf("%d\n", st.top());

10828

https://www.acmicpc.net/problem/10828

- empty(); 스택이 비어있는지 여부를 bool형 변수(true, false)를 반환하여 알 수 있다. Ex) if(st.empty()) printf("비어있음");
- size(); 현재 스택에 몇 개의 원소가 들어있는지 알 수 있다. Ex) printf("size : %d", st.size());
- 이 정도의 함수를 사용할 수 있다면 스택 관련 문제를 푸는 과정에서 막힘이 별로 없다.

10828

https://www.acmicpc.net/problem/10828

- C++ Code(github) : https://github.com/OfficialDominyellow/AlgorithmByDominyellow/blob/master/BackjoonOnlineJudge/10828.cc

키로거

5397

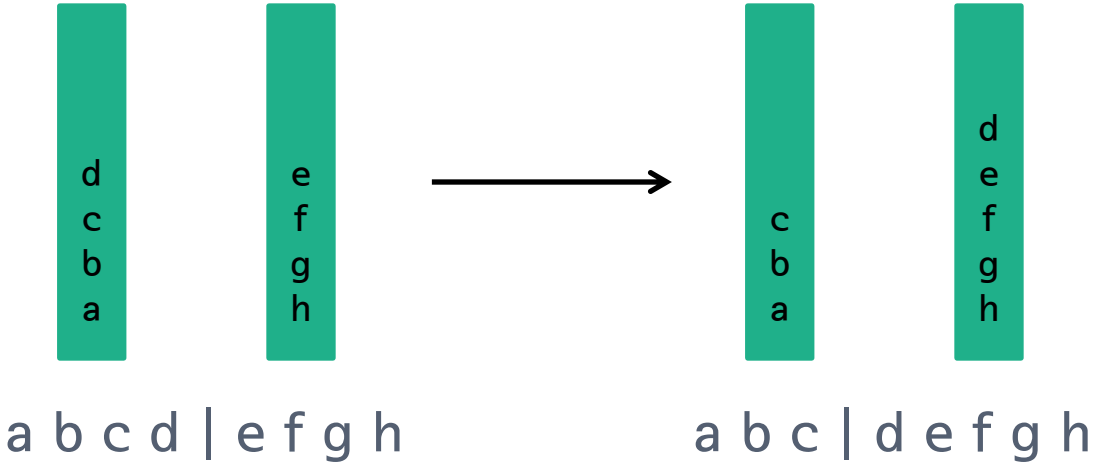
https://www.acmicpc.net/problem/5397

- 자료구조중 스택을 이용하여 풀 수 있는 문제이다.
- 스택, 큐를 이용하고 응용하여 해결하는 문제는 실전에서도 자주 나오기 때문에 항상 생각하고 있을 필요가 있다.
- 스택 2개를 이용하여 문제를 풀도록 한다.
- 현재 커서 위치의 왼쪽, 오른쪽을 나눠서 왼스택, 오른스택에 문자를 집어넣는다는 아이디어로 문제에 접근한다.

5397

https://www.acmicpc.net/problem/5397

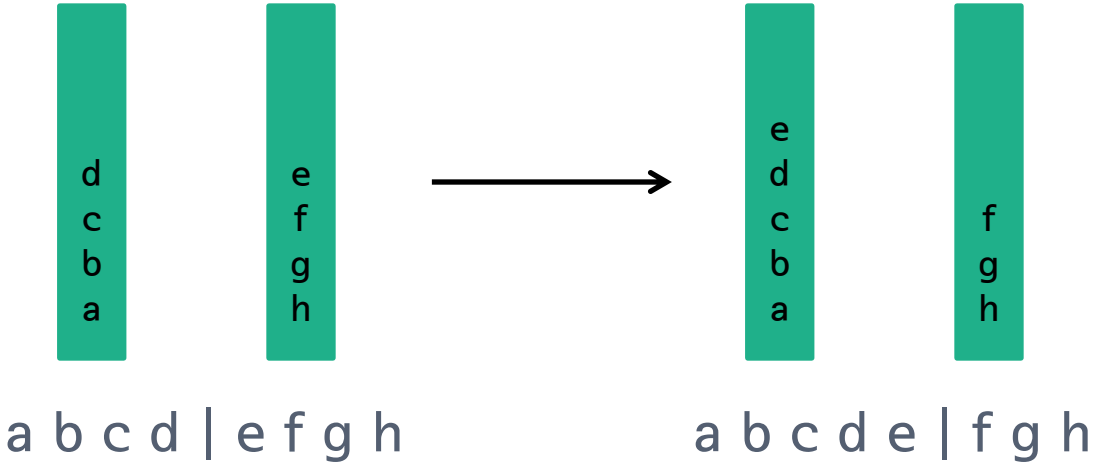
- '<' : 커서를 왼쪽으로 옮긴다. = 왼스택의 top원소를 pop하여 오른스택으로 push한다. 단, 왼스택이 비어있을 경우 아무일도 하지 않는다.



5397

https://www.acmicpc.net/problem/5397

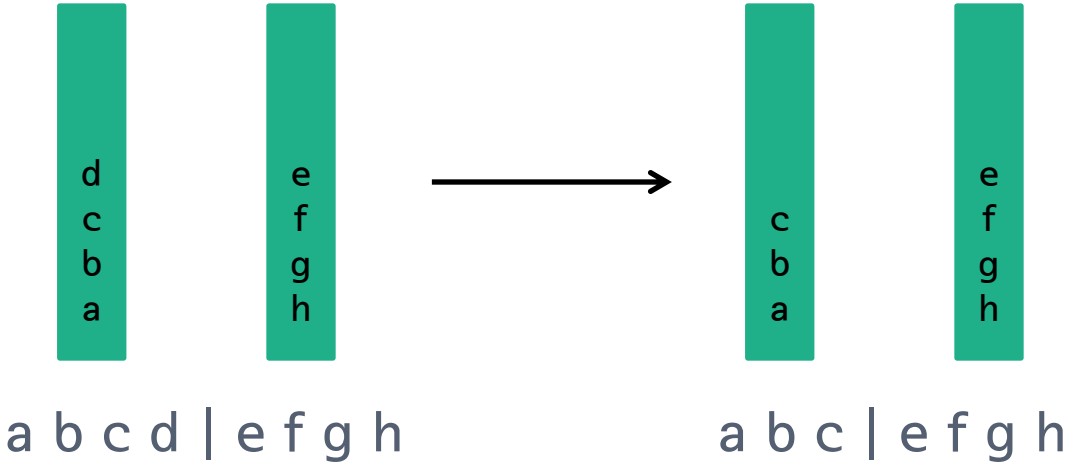
- '>' : 커서를 오른쪽으로 옮긴다. = 오른스택의 top원소를 pop하여 왼스택으로 push한다. 단, 오른스택이 비어있을 경우 아무일도 하지 않는다.



5397

https://www.acmicpc.net/problem/5397

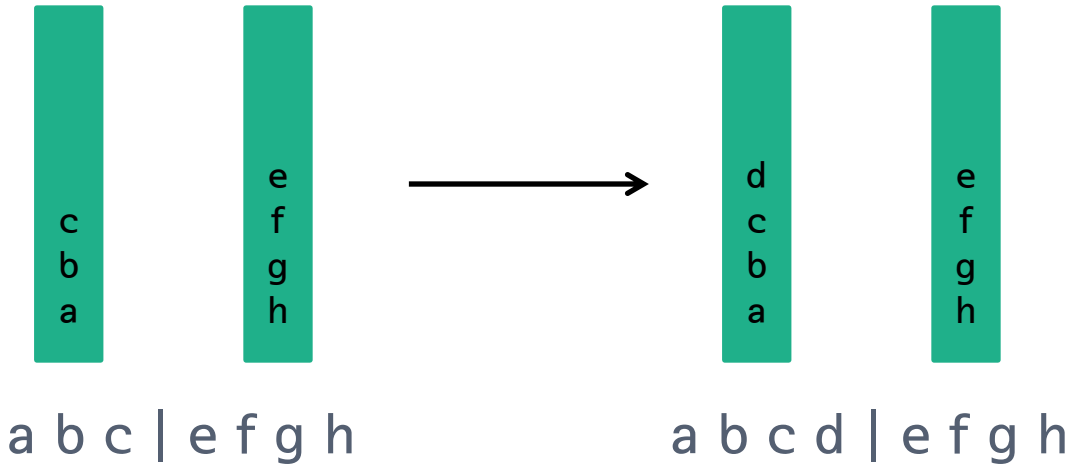
- '-' : 백스페이스(왼쪽 글짜 지우기). = 왼스택의 top원소를 pop한다. 단, 왼스택이 비어있을 경우 아무일도 하지 않는다.



5397

https://www.acmicpc.net/problem/5397

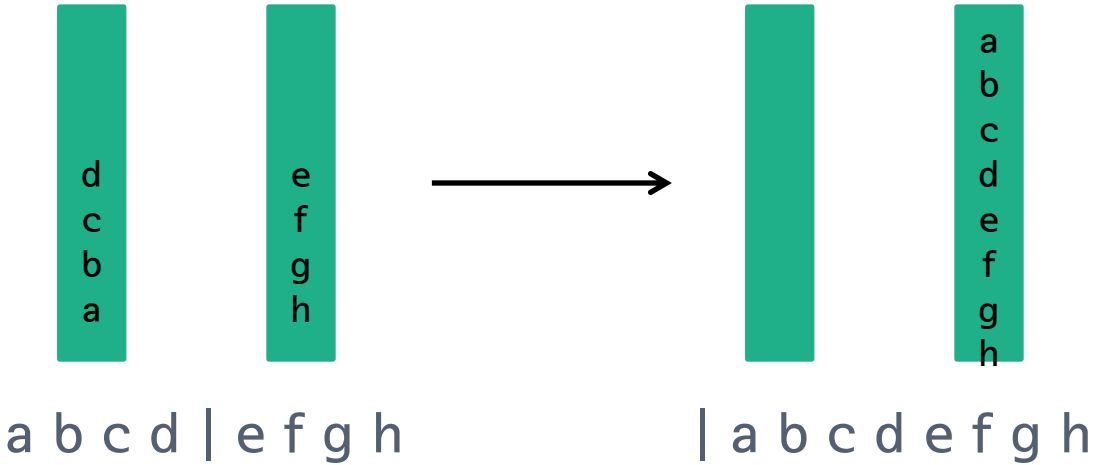
- ‘<‘, ‘>’, ‘-’ 이외의 문자 : 왼스택에 입력받은 문자를 push한다.



5397

https://www.acmicpc.net/problem/5397

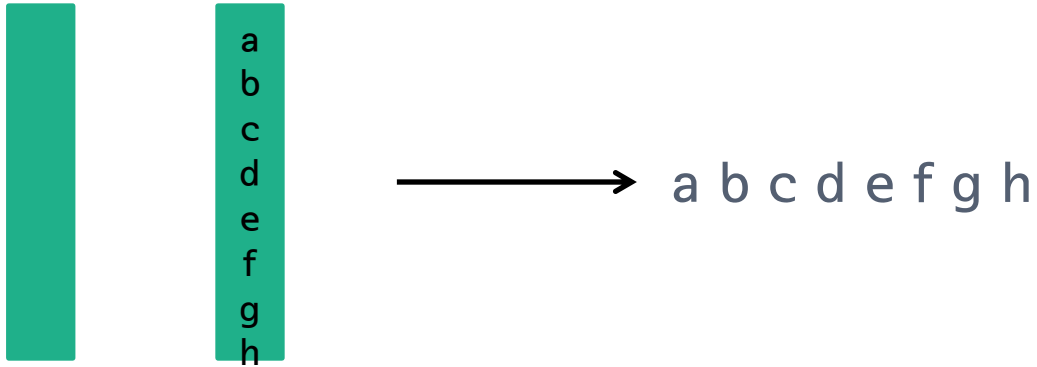
- 모든 키 입력을 마무리한 뒤 왼스택의 원소를 top부터 차례대로 pop 하여 오른스택으로 push 한다.



5397

https://www.acmicpc.net/problem/5397

- 오른스택이 빌 때 까지 원소를 pop하여 출력한다.



5397

https://www.acmicpc.net/problem/5397

- C++ Code(github) : https://github.com/OfficialDominyellow/AlgorithmByDominyellow/blob/master/BackjoonOnlineJudge/5397.cc

나는 요리사다

2953

https://www.acmicpc.net/problem/2953

- 배열을 이용한 단순 구현 문제이다.
- 5명의 요리사의 점수 4개가 차례대로 들어온다.
- 출력해야 할 것은 최고점수를 받은 요리사의 인덱스와 그 점수
- 1명의 점수 4개가 입력이 되면 합산하여 현재 점수 최댓값과 비교를 한다. 현재 최댓값보다 크면 값을 갱신하고 인덱스도 갱신한다.

2953

https://www.acmicpc.net/problem/2953

- C++ Code(github) : https://github.com/OfficialDominyellow/AlgorithmByDominyellow/blob/master/BackjoonOnlineJudge/2953.cc

소수 경로

1963

https://www.acmicpc.net/problem/1963

- 1부터 N까지 범위 안에 들어가는 모든 소수를 구하려면 에라토스테네스의 체를 사용한다.
- 2부터 N까지 모든 수를 써놓는다.
- 아직 지워지지 않은 수 중에서 가장 작은 수를 찾는다.
- 그 수를 지우고 소수로 저장한다.
- 이제 그 수의 배수를 모두 지운다.

1963

https://www.acmicpc.net/problem/1963

- 지워지지 않은 수 중에서 가장 작은 수는 2이다.
- 2는 소수이고 2의 배수를 모두 지운다.

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

1963

https://www.acmicpc.net/problem/1963

- 지워지지 않은 수 중에서 가장 작은 수는 2이다.

- 2는 소수이고 2의 배수를 모두 지운다.

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

1963

https://www.acmicpc.net/problem/1963

- 3의 배수를 지운다.

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

1963

https://www.acmicpc.net/problem/1963

- 5의 배수를 지운다.

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

1963

https://www.acmicpc.net/problem/1963

- 7의 배수를 지운다.

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

1963

https://www.acmicpc.net/problem/1963

만약 100까지의 수에서 소수를 구하려 한다면 이 단계에서 멈춘다.

11*11 > 100이기 때문에 더 이상 수행할 필요없이

남아있는 모든 수가 소수이다.

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

1963

https://www.acmicpc.net/problem/1963


```
#define MAX 10000
bool is_prime[MAX+1];

void erathos(){
    memset(is_prime, true, MAX+1);

    is_prime[0] = is_prime[1] = false;

    for(int i=2; i*i<MAX+1; i++){
        for(int j=i+i; j<MAX+1; j += i){
            if(is_prime[j]){
                is_prime[j] = false;
            }
        }
    }
}
```

1963

https://www.acmicpc.net/problem/1963

- 이후에 BFS를 이용하여 문제를 해결한다.

- 큐에는 현재 숫자와 첫 숫자부터의 소수거리를 묶어서 집어넣어야 한다. (구조체 혹은 pair를 이용하여 여러 개의 데이터를 묶어서 제어하는 것은 매우 중요하다!)

```
typedef struct PRIME{
    int num; //소수
    int dist; //거리
};
queue<PRIME> q;
```

```
queue<pair<int, int> > q
//first : 소수, second : 거리
```

1963

https://www.acmicpc.net/problem/1963


```
int bfs(){
    queue<pair<int, int> > q; //num, dist
    q.push(make_pair(F, 0));
    visited[F] = true;

    //목표 소수가 시작 소수와 같다면 0을 반환
    if(F == T){
        return 0;
    }

    int n, t;

    while(!q.empty()){

        n = q.front().first; //소수
        t = q.front().second; //거리
        q.pop(); //큐에서 원소를 pop

        int carry=1; //carrt : 올림수
...
```

```
        //4자리 숫자이기 때문에 각 자리수를 바꿔서 만들 수 있는 소수를 탐색
        for(int i=0; i<4; i++){
            int next;
            next = n-n/carry%10*carry;
            for(int j=0; j<10; j++){
                //숫자 1개만 바꿔서 방문한 적이 없는 소수를 만들 수 있다면
                if(is_prime[next] && !visited[next]){
                    q.push(make_pair(next, t+1)); //큐에 푸쉬 (현재 거리에 +1)
                    visited[next] = true; //방문 표시

                    //원하는 소수를 찾았을 시에 거리를 반환
                    if(next == T){
                        return t+1;
                    }
                }
                next += carry;
            }
            carry *= 10;
        }
    }
    return -1; //원하는 소수를 찾지 못하면 -1을 반환
}
```

1963

https://www.acmicpc.net/problem/1963

- C++ Code(github) : https://github.com/OfficialDominyellow/AlgorithmByDominyellow/blob/master/BackjoonOnlineJudge/1963.cc

트리의 지름

1967

https://www.acmicpc.net/problem/1967

- 노드의 개수가 10000개 이하이기 때문에 인접 배열을 통해서 graph를 추상화 할 수 없다. (10000 by 10000 2차원 배열 선언 불가능)
- 2차원 vector를 이용하여 인접List로 graph를 추상화 한다

1967

https://www.acmicpc.net/problem/1967


```
vector<vector<pair<int, int> > > graph; //[from] <to, value>

int main(void){
		scanf("%d", &N);

		graph.resize(10000+1);
		int f, t, v;
		for(int i=0; i<N-1; i++){
			scanf("%d %d %d", &f, &t, &v);
			graph[f].push_back(make_pair(t, v));
			graph[t].push_back(make_pair(f, v));
		}
}
```

1967

https://www.acmicpc.net/problem/1967

- BFS를 이용해서 특정 Node에서 가장 멀리있는 Node의 번호와 거리를 알 수 있다. (이정도 BFS는 할 줄 아는 상태에서 풀이를 볼 것)
- Key Idea!! : Tree에서 임의의 Node를 시작점으로 하여 가장 멀리있는 Node를 찾는다. 다시 그 Node를 시작점으로 하여 가장 멀리있는 Node를 찾으면 그 거리가 Tree의 지름이 된다.
- 즉, BFS 2번으로 Tree의 지름을 찾을 수 있다.

1967

https://www.acmicpc.net/problem/1967

- C++ Code(github) : https://github.com/OfficialDominyellow/AlgorithmByDominyellow/blob/master/BackjoonOnlineJudge/1967.cc

창영마을

3028

https://www.acmicpc.net/problem/3028

- 공은 처음에 1에 위치하고 있다.
- A : 공이 1, 2에 있으면 교환, 3에 있으면 아무 일도 일어나지 않는다.
- B : 공이 2, 3에 있으면 교환, 1에 있으면 아무 일도 일어나지 않는다.
- C : 공이 1, 3에 있으면 교환, 2에 있으면 아무 일도 일어나지 않는다.

3028

https://www.acmicpc.net/problem/3028

- 처음 문자열을 입력 받고 문자열의 길이만큼 반복문을 통해서 컵을 섞는 순서를 받아온다.

- scanf("%s", str); 과 같이 str에 문자열을 받아온 후에 아래처럼 두 가지 방법으로 for문을 작성할 수 있다.

1. for(int i=0; i<strlen(str); i++){};

2. Int len = strlen(str);
for(int i=0; i<len; i++){};

3028

https://www.acmicpc.net/problem/3028

1. for(int i=0; i<strlen(str); i++){};

2. Int len = strlen(str);
for(int i=0; i<len; i++){};

- strlen 함수는 시간 복잡도가 O(n)이다. 따라서 1번과 같은 방식으로 for문을 작성할 경우 언뜻 보기에는 시간복잡도가 O(n)이지만 사실 O(n*n)의 시간복잡도의 코드가 된다.
- 앞으로 for문 내부에 strlen함수를 쓰는 경우는 예외 없이 절대 지양 해야하는 방식이다.

3028

https://www.acmicpc.net/problem/3028

- C++ Code(github) : https://github.com/OfficialDominyellow/AlgorithmByDominyellow/blob/master/BackjoonOnlineJudge/3028.cc

기숙사 바닥

2858

https://www.acmicpc.net/problem/2858

- 직사각형을 만들 수 있는 모든 방식을 탐색하여 해결한다.
- 먼저 B를 직사각형으로 만들 수 있는 모든 가로, 세로를 찾는다. (가로 >= 세로 일 때만 탐색)
- i를 1부터 I <= sqrt(B) 까지 증가 시킨다. i X [B/i]가 B와 같아지면 갈색 타일을 i와 [B/i]를 가로와 세로로 하는 직사각형을 만들 수 있다는 것이다.

2858

https://www.acmicpc.net/problem/2858

- 갈색 타일의 가로, 세로가 결정되면 갈색 타일의 가로와 세로에 2씩 더하면 빨간색 타일을 포함한 전체 직사각형의 가로와 세로 길이를 얻어낼 수 있다.
- L = [B/i] + 2;, W = i+2;
- 이 때 (L+W-2)X2가 R과 같다면 직사각형을 만들 수 있다.
- 해가 유일한 경우만 input으로 주어진다고 했기 때문에 L, W를 출력하고 프로그램을 종료하면 된다.

2858

https://www.acmicpc.net/problem/2858

- C++ Code(github) : https://github.com/OfficialDominyellow/AlgorithmByDominyellow/blob/master/BackjoonOnlineJudge/2858.cc