목 차

문법 및 함수 정리

for문 매크로

```
#define FOR(x,n,m)   for(int x=n;x<(m);x++)
```

lower_bound, upper_bound

lower_bound(first, last, val);
-> first부터 last까지 중 val 보다 크거나 같은 값의 첫 위치
upper_bound(first, last, val);
-> first부터 last까지 중 val 보다 큰 값의 첫 위치

C++ 구조체 생성자 오버로딩

```
typedef struct S{
    int x,y,z;
    S(){z=1;}
    S(int X, int Y, int Z) : x(X),y(Y),z(Z){}
};
```

위처럼 구조체도 생성자 오버로딩이 가능함.

map 순회

```
for(auto it=m.begin(); it!=m.end(); it++)
```

해당 위치의 데이터 값 접근

```
it->first, it->second
```

2차원 배열 90도 회전

```
int temp_arr[100][100];
FOR(i,0,n) FOR(j,0,n) temp_arr[i][j] = arr[n-j-1][i];
memmove(arr,temp_arr,sizeof(arr));
```

2차원 행,열 바꾸기

```
int temp_arr[100][100];
FOR(i,0,n) FOR(j,0,n) temp_arr[i][j] = arr[j][i];
memmove(arr,temp_arr,sizeof(arr));
```

배열 내 최소, 최대값 구하기

*min_element(zSum, zSum+5);

*max_element(zSum, zSum+5);

배열 내 최소,최대를 구하는 함수로써 비교 함수도 인자로 넣어줄 수 있다.

주의할 점은 배열의 return 값이 배열의 주소 값이기 때문에 *을 붙여줘야한다.

틀린 문제

백준 16235 - 나무 재테크

-> 단순 simulation 문제
-> 각각의 나무를 vector의 2차원 배열로 표현했음.
-> 봄 계절에 양분을 얻지 못하는 나무는 죽은 나무로 표현하고 삭제하지 않음.
-> 이유는 vector의 erase를 쓰고 싶지 않았기 때문에.
-> 하지만 데이터를 남기고 있던 것이 오히려 시간을 더 오래 걸리게 했음.
-> vector의 pop_back() 함수를 이용하면 해결가능함.
-> 원래는 pop_back을 안써서 vector의 뒤에서부터 탐색하며 erase를 했는데 대안이 생김.

백준 17143 - 낚시왕

-> simulation 문제
-> 상어 잡기, 상어 움직이기 모두 구현했는데, 상어가 겹칠 때 효율적으로 삭제시키는 방법을 알아야함.
-> 구현은 할 수 있겠지만 너무 비효율적인 방법인 것 같음.
-> 삭제할 때 해당 좌표에 상어들을 다 넣어놓고 나중에 비교하고 삭제하려고 했음.
-> 이렇게 하면 가장 큰 상어를 구할지라도 삭제하는 과정에서 vector의 erase를 사용하고 코드가 길어짐.
-> 구조체 S 타입으로 2차원 배열 선언하고 상어의 이동이 끝날 때마다 해당 위치 상어와 비교
-> 크기가 크다면 덮어 씌우고 크기가 작다면 다음으로 넘어감.

맞은 문제

백준 14890 - 경사로
-> 단순 simulation 문제
-> 경사로를 넣는 조건이 다양해서 조금 까다로웠음.
-> 행, 열에 대해 답을 구해야 하는데 이런 경우 2차원 배열을 90도 돌리는 방법을 이용하면 쉬워짐.

백준 15684 - 사다리 조작
-> 단순 simulation 문제 + dfs
-> 문제 자체는 어려워 보이지 않는데 구현이 귀찮았음.
-> 히든케이스(사다리를 놓지 않아도 통과되는 경우를 생각 못함.
-> 코드 너무 더럽게 짰음. 숏코딩 참고하자.
-> 꼭 다시 풀어보기.

백준 14891 - 톱니바퀴
-> 단순 simulation 문제
-> 돌리는 바퀴 기준으로 방향 변수에 대해 실수함.

백준 15686 - 드래곤커브
-> 단순 simulation 문제
-> 지금까지 만들어진 방향을 vector에 저장
-> 1세대를 그릴 때마다 vector의 역순으로 탐색.
-> 탐색을 하면서 각 방향에 +1을 해주고 그려줌.
-> Stack을 사용할까 생각했지만 vector로도 충분히 가능할 것 같아서 vector로 구현함.
-> 생각해보니 Stack은 탐색이 힘들어서 적합하지 않을 듯.

백준 16236 - 아기 상어
-> bfs의 반복 문제
-> 먹이를 찾았다고 bfs를 끝내면 안되고 해당 깊이까지는 다 돌아야함.
-> 해당 깊이에서 후보군에 넣을 때 조건이 필요함(0이 아니고 자신보다 작은 크기일 때)
-> bfs 순회 중 현재 깊이가 먹이를 먹은 깊이보다 많을 때 업데이트하고 return 시킴.
-> 이러면 먹이를 먹고 다음 깊이가 없다면 조건에 걸리지 않아 업데이트가 안됨.
-> 업데이트를 while문 밖에 두면 됨.
-> 그러면 조건에 걸려서 break되거나 더이상 순회할 좌표가 없을 때도 업데이트가 됨.
-> 단, 먹이를 먹을 때 값이 변하는 dd 값이 -1이라면 먹이를 못 먹은 경우이므로 return false.

백준 17142 - 연구소 3
-> 조합 찾기 후 bfs 문제
-> 처음에 단순히 bfs가 끝나면 답을 업데이트 시켰음.
-> 이런 경우 빈칸이 다 없어져도 활성화되지 않은 바이러스도 활성화 시키느라 오답이 됨.
-> bfs while에 들어가기 전에 빈칸의 개수를 세고 0개가 되면 while 탈출하도록 함.
-> 조합 찾을 때 dfs로 했는데 return 조건에서 실수함.

백준 17140 - 이차원 배열과 연산

-> simulation 문제
-> 배열을 검사해서 새로운 배열을 만드는 것은 구현했음.
-> 하지만 시험장에서 풀었다면 시간이 부족했을 것 같음.
-> 행마다 검사, 열마다 검사를 각각 따로 만들려고 했기때문.
-> 배열을 90도 돌리는 방법도 생각했지만 배열 모양이 정사각형이 아니기 때문에 불가능.
-> 생각해보니 회전이 아니라 행,열을 바꾸면 되는 문제였음.
-> 계산하는 함수는 1개만 만들고 행,열을 바꿔주며 계속 진행함.

백준 17779 - 게리맨더링2

-> simulation 문제
-> 처음에 구역을 어떻게 나눌지에 대해 고민함.
-> 구역을 모두 5로 채운 다음 각 지역에 맞게 채우면 됨.
-> 1,3 구역은 열 0부터 시작해서 경계선에 만나면 다음 행으로 넘어감.
-> 2,4 구역은 열 n-1부터 시작해서 경계선에 만나면 다음 행으로 넘어감.