Queue : 우선 순위

내가 짠 코드

```
#include <iostream>
#include <vector>
#include <queue>
#include <algorithm>
using namespace std;

int solution(vector<int> priorities, int location) {
int answer = 0;
queue<int> q;
int temp=0;
int index = 0;

for (int p : priorities) {
    q.push(p);
}

//벡터 정렬
sort(priorities.begin(), priorities.end());
int max = priorities[index];

while (!q.empty()) {

    if(q.front()<max)
    {
        temp = q.front();
        q.push(temp);
    }
    else if( q.front() == max)
    {
        answer++;
    }
```

```
        else if(q.front() == max && location == 0)
        {
            answer++;
            break;
        }
        q.pop();
        if(location==0)
        {
            location = q.size() - 1;
        }
        location--;


    }

    return answer;
    }

    int main()
    {
    vector<int> priorities = { 2,1,3,2 };
    int location = 2;
    int answer = solution(priorities, 2);
    cout << answer;
    }
```

정답

```

#include <iostream>
#include <vector>
#include <queue>
#include <algorithm>
using namespace std;

int solution(vector<int> priorities, int location) {
    int answer = 0;
    queue<pair<int, int>> q; // (우선순위, 위치) 쌍을 저장하는 큐
    int N = priorities.size();

    for (int i = 0; i < N; ++i) {
        q.push({priorities[i], i});
    }

    sort(priorities.begin(), priorities.end(), greater<int>
()); // 내림차순 정렬

    int idx = 0;
    while (!q.empty()) {
        int prio = q.front().first;
        int loc = q.front().second;
        q.pop();

        if (prio == priorities[idx]) {
            ++answer;
            ++idx;

            if (loc == location) {
                break; // 찾고자 하는 문서가 출력되었으므로 종료
            }
        } else {
            q.push({prio, loc}); // 우선순위가 아니면 다시 큐에
넣음
        }
    }

    return answer;
}
```