5주차: 큐(Queue)

1. 개념
    ▼ FIFO(First In First Out)의 자료구조
        - 들어온 순서로 나가기(줄서기, 요세푸스 문제 등)에 활용
        - BFS(Breadth First Search, 너비우선탐색)에 활용
    ▼ Queue의 ADT
        ▼ 연산
            - boolean isFull(): 큐에 들어 있는 데이터 캐수가 maxsize인지 확인해 boolean값을 반환
            - boolean isEmpty(): 큐에 들어 있는 데이터가 없는지 확인해 boolean값을 반환
            - void push(ItemType Time): 큐에 데이터 푸쉬
            - ItemType pop(): 큐에서 처음 푸쉬한 데이터를 팝하고 그 데이터 반환
        ▼ 상태
            - Int front: 큐에서 가장 마지막에 팝한 위치 기록
            - Int rear: 큐에서 가장 최근에 푸쉬한 데이터 위치 기록
            - ItemType data[maxsize]: 큐의 데이터 관리하는 배열. 최대 maxsize개 데이터 관리
2. 사용 예시
    ▼ Queue 사용 예시

```
/*
# Queue(FIFO) 사용
- std::queue
- 데이터를 순서대로 처리해야 할 때(BFS, 대기열)
- 중간요소에 직접 접근 불가.
*/

#include <iostream>
#include <queue>
```

```
using namespace std;

int main() {
  queue<int> q;
  q.push(1);
  q.push(2);
  q.push(3);
  cout << "Front element: " << q.front() << endl;

  q.pop();
  cout << "Queue element after pop: " << q.front() << e

  if (!q.empty()) {
    cout << "Queue is not empty" << endl;
  }

  cout << "Queue size: " << q.size() << endl;
  return 0;
}
```

▼ Queue적용 문제: 요세푸스

```
#include <iostream>
#include <queue>

using namespace std;

int josephus(int N, int K){
    queue<int> q;
    for (int i=1;i<=N;++i){
        q.push(i);
    }
    while (q.size()>1){
        for (int i=0; i<K-1; ++i){
            q.push(q.front())
            q.pop()
    }
```

```
    return q.front();
}

int main(){
    int N=5;
    int K=2;
    cout << "The survivor is: " << josephus(N,K) << end
    return 0;
}
```

3. 문제

▼ 기능개발

```
#include <vector>
#include <cmath>
using namespace std;

vector<int> solution(vector<int> progresses, vector<int
    vector<int> answer;
    int n = progresses.size();
    vector<int> days_left(n);

    for (int i = 0; i < n; ++i) {
        days_left[i] = ceil((100.0 - progresses[i]) / s
    }

   int count = 0;
    int max_day = days_left[0];


    for (int i = 0; i < n; ++i) {
        if (days_left[i] <= max_day) {
            count++;
        } else {
            answer.push_back(count);
            count = 1;
            max_day = days_left[i];
        }
```

```
    }

    answer.push_back(count);
    return answer;
}


/*Test*/
#include <iostream>
#include <iterator>

void print(vector<int> vec)
{
    copy(vec.begin(), vec.end(), std::ostream_iterator<
    cout << endl;
}

int main()
{

    print(solution({93, 30, 55}, {1, 30, 5})); // 2 1
    print(solution({95, 90, 99, 99, 80, 99}, {1, 1, 1, 

    return 0;

}
```

▼ 카드뭉치

```
#include <iostream>
#include <queue>
#include <string>
#include <vector>

using namespace std;

string solution(vector<string> cards1, vector<string> c
    queue<string> c1, c2, g;
```

```
    for (const string& s : cards1) c1.push(s);
    for (const string& s : cards2) c2.push(s);
    for (const string& s : goal) g.push(s);

    while (!g.empty()) {
        if (!c1.empty() && c1.front() == g.front()) {
            c1.pop();
            g.pop();
        }
       else if (!c2.empty() && c2.front() == g.front())
            c2.pop();
            g.pop();
        }
       else {
            break;
        }
    }
    return g.empty() ? "Yes" : "No";
}



/*Test*/
#include <iostream>

int main()
{
    cout<< solution({"i", "drink", "water"}, {"want", "
    cout<< solution({"i", "water", "drink"}, {"want", "
    return 0;

}
```