과제 #1 센서 네트워크 - 201921438 조현태

1) 소스코드

```
#include <stdio.h>
#include <iostream>
#include <algorithm> // sort를 위한 헤더
#include <math.h> // pow를 위한 헤더
using namespace std;

// -20000 <= x <= 20000이므로 배열의 크기 40001
int North[40001];
// N <= 100000
int South[100000];

int sum[40001];
int answer[100000];

int main(void)
{
	int d, r, n; // d = 강폭의 넓이, r = 레이더 반경, n = 레이더 개수
	cin >> d >> r >> n;

	// 북쪽의 레이더 -> 배열의 인덱스와 x좌표를 동일시
	for (int i = 0; i < n; i++)
	{
		int a;
		cin >> a;
		North[a + 20000] += 1;
	}

	// 남쪽의 레이더 -> x좌표를 순차적을 받음 (-20000~20000 ==> 0~4000)
	for (int i = 0; i < n; i++)
	{
		int a;
		cin >> a;
		South[i] = a + 20000;
	}

	if (d <= r)
	{
		// 피타고라스정리에 의하여 x좌표의 허용치 구하기 -> d = 5, r = 7 -->
int(pow(24, 0.5)) > 4 --> int => 4
		int deadline = int(pow(pow(r, 2) - pow(d, 2), 0.5)); // 인덱스에 넣기 위해
```

서 int처리

```
                // 구간합 알고리즘 사용
                for (int i = 0; i < 40001; i++)
                {
                        if (i == 0) { sum[i] = North[i]; }
                        else { sum[i] = sum[i - 1] + North[i]; }
                }

                // 구간 안의 안테나의 개수 구하기
                for (int i = 0; i < n; i++)
                {
                        // 구간 만들기
                        int left = South[i] - deadline;
                        int right = South[i] + deadline;

                        // 구간이 인덱스 범위를 넘어가는 경우
                        if (right > 40000) { right = 40000; }
                        if (left <= 0)
                        {
                                answer[i] = sum[right]; // -20000이하의 범위는 무조건 0
으로 처리.
                        }
                        else
                        {
                                answer[i] = sum[right] - sum[left - 1];
                        }
                }
                sort(answer, answer + n);

                // 정답 출력 (정수 n개, 빈칸 n-1개 이므로)
                cout << answer[0];
                for (int i = 1; i < n; i++)
                {
                        cout << " " << answer[i];
                }
                cout << endl;
        }
        else  // d > r이면 범위내 레이더가 있을 수 없으므로 0을 출력하고 프로그램 종료.
        {
                cout << 0;
                for (int i = 1; i < n; i++)
                {
```

```
                cout << " " << 0;
            }
            cout << endl;
        }
        return 0;
}
```

2) 문제 설명

남쪽 레이더의 통신반경을 기준으로 통신반경 안의 북쪽 레이더 개수를 구하는 문제이므로
북쪽 레이더 배열(North)에는 인덱스를 x좌표화 해서 해당 x좌표 위치에 개수만 입력받고
(-20000 ~ 20000이므로 North[0] ~ North[40000]로 표현한 후, 해당 인덱스에 +1을 함.)
남쪽 레이더 배열(South)에는 n개의 x좌표를 직접 받습니다. (x좌표한 인덱스를 이용.)

```
int North[40001]; , int South[100000];
for (int i = 0, i < n, i++) {North[x+20000] += 1;}
for (int i = 0, i < n, i++) {South[i] = x+20000; }
```

이후 강폭의 넓이(d)와 레이더반경(r)을 이용해서 피타고라스정리를 통해
x좌표에 대한 범위(deadline)를 구합니다. (인덱스는 정수이므로 값을 int로 바꿈.)

```
int deadline = int(pow(pow(r, 2) - pow(d, 2), 0.5));
```

그리고 North배열을 통해 레이더 개수에 대한 구간합(sum) 배열을 구한 후,
deadline을 이용해 좌우 구간(left, right)을 설정합니다.

```
int sum[40001];
for (int i = 0; i < 40001; i++)
            {
                if (i == 0) { sum[i] = North[i]; }
                else { sum[i] = sum[i - 1] + North[i]; }
            }
int left = South[i] - deadline;
int right = South[i] + deadline;
```

구간합 배열을 이용해 sum(right) - sum(left-1)로 South배열의 레이더에 대한
구간 안의 North 레이더의 개수를 구합니다. (범위를 벗어나는 경우는 따로 처리함.)

```
int answer[100000];
answer[i] = sum[right] - sum[left - 1];
```

이후 오름차순으로 정렬하여 출력합니다.

```
sort(answer, answer + n);
```