

(훈련반1) Level23.5

가지치기를 얼마나 잘 하느냐에 따라서 재귀함수의 동작 속도는 더 빨라질 수 있습니다.

그리고 via 전역배열은 앞으로 배울 DFS 알고리즘에 활용되는 방법입니다.

Trace를 통해 개념을 확실하게 잡고 넘어가주시면 됩니다.

Level23.5 왼쪽, 오른쪽 이동 [난이도 : 2]

문제 1번 [숙제 목록보기]

3	5	1	9	7
---	---	---	---	---

위 배열을 하드코딩 해주세요.

그리고 R 또는 L 문자 4개를 입력 받습니다.

R은 right 방향을 의미하고

L은 left 방향을 의미 합니다.

아래 그림과 같이

R을 입력 받으면 숫자를 오른쪽으로 한칸씩 모두 이동시키는데 맨 뒤에 있는 숫자는 맨앞으로 와야합니다.



반대로 L을 입력 받으면 숫자를 왼쪽으로 한칸씩 모두 이동 시키고 맨 앞에 있는 숫자는 맨 뒤로 보냅니다.



R 또는 L을 4번 입력 받은 후 처리된 결과를 출력 해주세요.

입력 예제

R

R

R

L

출력 결과

9 7 3 5 1

```

#include <iostream>
using namespace std;

const int arrSize = 5;
const int inputSize = 5; // 값 4개 입력받음

int main(void)
{
    int arr[arrSize] = { 3,5,1,9,7 };
    char input[inputSize] = {}; // L 또는 R 총 4개 입력
    cin >> input;

    int temp[arrSize] = {};
    for (int i = 0; i < inputSize; ++i)
    {
        if (input[i] == 'R')
        {
            for (int k = 0; k < arrSize; ++k)
            {
                if (k == 4)
                {
                    temp[0] = arr[4];
                }
                else
                {
                    temp[k + 1] = arr[k];
                }
            }
        }
        else
        {
            for (int o = arrSize - 1; o >= 0; --o)
            {
                if (o == 0)
                {
                    temp[4] = arr[0];
                }
                else
                {
                    temp[o - 1] = arr[o];
                }
            }
        }

        for (int m = 0; m < arrSize; ++m)
            arr[m] = temp[m];
    }
}

```

```
    for (int n = 0; n < arrSize; ++n)
        cout << arr[n] << " ";

    return 0;
}
```

Level23.5 암살자 존획 [난이도 : 3]

문제 2번 [숙제 목록보기]

#은 암살자들이 있는 위치 입니다. 3명의 암살자의 위치를 입력 받으세요.

만약, 직선거리에 상대방이 있다면 서로 총을 쏘게 됩니다.

안전한 배치



#			
		#	
	#		

서로 총을 쏘는
위험한 배치



#			#
	#		

세명의 좌표를 입력 받고

서로 총을 쏘지 않는 안전한 위치라면 "안전" 출력

그렇지않다면 "위험"을 출력 해주세요.

입력 예제

0 0

1 2

2 1

출력 결과

안전

```

#include <iostream>
using namespace std;

const int sizeY = 3;
const int sizeX = 4;
const int inputSize = 3;

int main(void)
{
    //char map[sizeY][sizeX] = {};
    int arrX[inputSize] = {};
    int arrY[inputSize] = {};
    for (int i = 0; i < inputSize; ++i)
        cin >> arrY[i] >> arrX[i];

    int dangerLineY[sizeY] = {};
    int dangerLineX[sizeX] = {};

    bool isDanger = false;
    for (int k = 0; k < inputSize; ++k)
    {
        int valY = arrY[k];
        int valX = arrX[k];

        if (dangerLineY[valY] == 0 && dangerLineX[valX] == 0)
        {
            ++dangerLineY[valY];
            ++dangerLineX[valX];
        }
        else
        {
            isDanger = true;
            break;
        }
    }

    if (isDanger)
        cout << "위험";
    else
        cout << "안전";

    return 0;
}

```

Level23.5 네모네모 더하기 [난이도 : 3]

문제 3번 [숙제 목록보기]

4x4 배열을 만들고 (0,0)~(2,2)까지 3 x 3칸에 다가 아홉 숫자를 입력 받으세요.

예를들어


1 2 1

2 3 4

3 2 1 을 입력 받았다면 아래와 같이 배열값이 놓여지게 됩니다.

그리고 빈칸에는 가로줄의 합& 세로줄의 합 & 대각선줄의 합이 계산되어 채워 집니다.

1	2	1	
2	3	4	
3	2	1	



1	2	1	4
2	3	4	9
3	2	1	6
6	7	6	5

배열에 모든 값이 채워지면 출력합니다.

적절한 for문을 사용하여 이 프로그램을 만들어 주세요

입력 예제

1 2 1

2 3 4

3 2 1

출력 결과

1 2 1 4

2 3 4 9

3 2 1 6

6 7 6 5

```
#include <iostream>
using namespace std;

const int arrSize = 4;
const int inputSize = 3;

int main(void)
{
    int map[arrSize][arrSize] = {};
    for (int y = 0; y < inputSize; ++y)
    {
        for (int x = 0; x < inputSize; ++x)
            cin >> map[y][x];
    }

    for (int i = 0; i < arrSize; ++i)
    {
        if (i == arrSize - 1)
        {
            for (int n = 0; n < inputSize; ++n)
            {
                map[i][i] += map[n][n];
            }
        }
        else
        {
            for (int m = 0; m < inputSize; ++m)
            {
                map[i][arrSize - 1] += map[i][m];
                map[arrSize - 1][i] += map[m][i];
            }
        }
    }

    for (int y = 0; y < arrSize; ++y)
    {
        for (int x = 0; x < arrSize; ++x)
            cout << map[y][x] << " ";
        cout << endl;
    }

    return 0;
}
```

Level23.5 숫자 transformation

[난이도 : 4]

문제 4번 [[숙제](#) [목록보기](#)]

3	5	4	1
1	1	2	3
6	7	1	2

위 숫자들을 하드코딩 해주세요.

그리고 각기 다른 숫자 4개를 배열에 입력 받으세요.

예로들어 **1 3 7 6** 을 입력 받았다고 한다면, 이차배열의 값을

숫자 1을 3으로 변경

숫자 3을 7로 변경

숫자 7을 6으로 변경

숫자 6을 1로 변경

하시면 됩니다.

(이 외 나머지 숫자는 그대로 두시면 됩니다)

변경된 이차배열 값을 출력해주세요.

ex)

1 3 7 6



7	5	4	3
3	3	2	7
1	6	3	2

입력 예제

1 3 7 6

출력 결과

7 5 4 3

3 3 2 7

1 6 3 2

```

#include <iostream>
using namespace std;

const int sizeY = 3;
const int sizeX = 4;
const int inputSize = 4;

int main(void)
{
    int map[sizeY][sizeX] =
    {
        3,5,4,1,
        1,1,2,3,
        6,7,1,2
    };
    int temp[sizeY][sizeX] = {};

    for (int y = 0; y < sizeY; ++y)
    {
        for (int x = 0; x < sizeX; ++x)
        {
            temp[y][x] = map[y][x];
        }
    }

    int input[inputSize] = {};
    for (int i = 0; i < inputSize; ++i)
        cin >> input[i];

    for (int k = 0; k < inputSize; ++k)
    {
        for (int y = 0; y < sizeY; ++y)
        {
            for (int x = 0; x < sizeX; ++x)
            {
                if (map[y][x] == input[k])
                {
                    if (k == inputSize - 1)
                        temp[y][x] = input[0];
                    else
                        temp[y][x] = input[k + 1];
                }
            }
        }
    }
}

```

```
for (int y = 0; y < sizeY; ++y)
{
    for (int x = 0; x < sizeX; ++x)
    {
        map[y][x] = temp[y][x];
    }
}

for (int y = 0; y < sizeY; ++y)
{
    for (int x = 0; x < sizeX; ++x)
        cout << map[y][x] << " ";
    cout << endl;
}

return 0;
}
```

Level23.5 자기자리 찾기 [난이도 : 6]

문제 5번 [숙제 목록보기]

8개의 숫자를 배열에 입력받아주세요.

배열에서 가장 왼쪽에 있는 숫자를 "피벗"이라고 합니다.

만약 아래와 같이 4 1 7 9 6 3 3 6을 입력받으면 피벗은 4가 됩니다.

4	1	7	9	6	3	3	6
---	---	---	---	---	---	---	---

이 배열을 가지고 아래에 나와있는 규칙대로 숫자들을 옮기다보면,

신기하게도

피벗 왼쪽에는 피벗보다 작은 숫자들이

피벗 오른쪽에는 피벗보다 큰 숫자들로 구성됩니다.

ex)

3	1	3	4	6	9	7	6
---	---	---	---	---	---	---	---

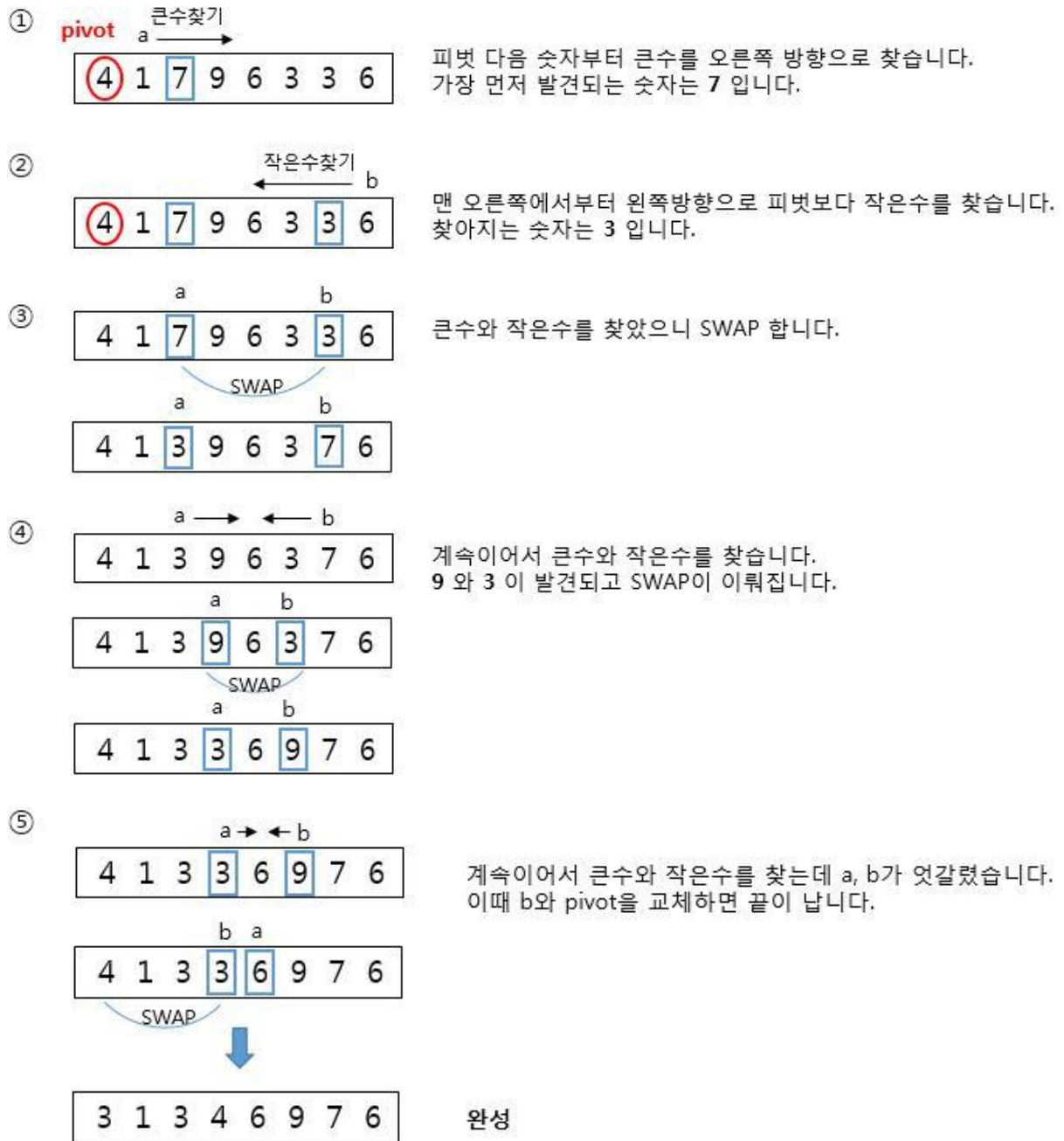
 적합한 위치로 이동 성공

아래의 규칙에 따라 숫자를 옮기고, 결과를 출력 해 주세요

요약 :

a는 pivot보다 큰수를 찾아야하고

b는 pivot보다 작은수를 찾아서 SWAP 해야합니다.



입력 예제

4 1 7 9 6 3 3 6

출력 결과

3 1 3 4 6 9 7 6

```

#include <iostream>
using namespace std;

const int inputSize = 8;

int main(void)
{
    int input[inputSize] = {};
    for (int i = 0; i < inputSize; ++i)
        cin >> input[i];

    int pivot = input[0];
    int idxA = 1;
    int idxB = inputSize - 1;

    int gap = idxB - idxA;
    while (gap > 0)
    {
        for (int m = inputSize - 1/*idxB*/; m >= 0; --m)
        {
            if (input[m] < pivot)
            {
                idxB = m;
                break;
            }
        }

        for (int n = 1; n < inputSize; ++n)
        {
            if (input[n] > pivot)
            {
                idxA = n;
                break;
            }
        }

        gap = idxB - idxA;

        if (gap > 0)
        {
            int temp = input[idxA];
            input[idxA] = input[idxB];
            input[idxB] = temp;
        }
        else
    }
}

```

```
        {  
            int temp = input[idxB];  
            input[idxB] = input[0];  
            input[0] = temp;  
        }  
    }  
  
    for (int k = 0; k < inputSize; ++k)  
        cout << input[k] << " ";  
  
    return 0;  
}
```

Level23.5 황금좌표 찾기 [난이도 : 5]

문제 6번 [숙제 목록보기]

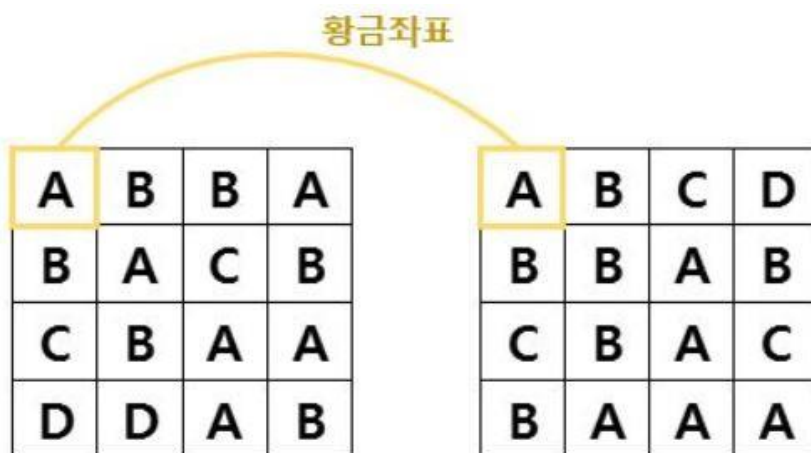
아래 그림을 보면 두개의 4x4 배열이 있습니다.

왼쪽배열 (4x4)는 입력받고,

오른쪽배열 (4x4)는 하드코딩 해 주세요.

이 두 배열에서 같은 좌표값이 같은 알파벳을 가지고 있으면 황금 좌표 입니다.

황금 좌표를 가장 많이 가진 알파벳을 찾아 출력 해주세요.



정답 : B

위 예제에서는 B가 황금좌표를 4개를 가지고 있기 때문에

정답은 B입니다.

입력 예제

ABBA

BACB

CBAA

DDAB

출력 결과

B


```

#include <iostream>
using namespace std;

const int arrSize = 4;
const int offset = 65;

int main(void)
{
    char arrL[arrSize][arrSize] = {};
    for (int y = 0; y < arrSize; ++y)
    {
        for (int x = 0; x < arrSize; ++x)
        {
            cin >> arrL[y][x];
        }
    }

    char arrR[arrSize][arrSize + 1] =
    {
        "ABCD",
        "BBAB",
        "CBAC",
        "BAAA"
    };

    // ascii -> A : 65
    int goldCnt[5] = {};
    for (int y = 0; y < arrSize; ++y)
    {
        for (int x = 0; x < arrSize; ++x)
        {
            if (arrL[y][x] == arrR[y][x])
            {
                int idx = arrL[y][x] - offset;
                ++goldCnt[idx];
            }
        }
    }

    int maxVal = goldCnt[0];
    int maxIdx = 0;

```

```
    for (int i = 0; i < 5; ++i)
    {
        if (maxVal < goldCnt[i])
        {
            maxVal = goldCnt[i];
            maxIdx = i;
        }
    }

    char result = maxIdx + offset;
    cout << result;
    return 0;
}
```