**Introduction to Computer and Lab**

**Homework #9**

**Due date: Jun 12, 2016**

**학번: 201404051**

**이름: 정 용 석**

**1. 행렬의 합을 출력**

**1.1 Solution**

일단 모든 문제의 행렬의 가로 세로의 최대 크기는 10이기 때문에 MAX\_SIZE를 10으로 두고, 최대 크기를 가지는 2차원 행렬 2개를 만들었다. 그리고 첫 번째 행렬에는 1부터 m\*n까지 값을 순차적으로 가지도록 대입하고, 두 번째 행렬은 2씩 증가하는 홀 수의 값들을 순차적으로 대입하였다. 마지막으로 출력과 동시에 첫 번째와 두 번째의 행렬 각각의 행과 열에 원소들을 더함과 동시에 출력했다.

**1.2. Source code**

void arraySum() // 행렬의 합 출력

{

int m, n; //m: 행, n: 열

int setA[MAX\_SIZE][MAX\_SIZE]; //행렬 A

int setB[MAX\_SIZE][MAX\_SIZE]; //행렬 B

int value = 1;

int i, j;

scanf("%d %d", &m, &n);

//행렬 A에는 1~m\*n을 차례대로 대입

for (i = 0; i < m; i++)

for (j = 0; j < n; j++)

setA[i][j] = value++;

value = 1;

//행렬 B에는 1부터 +2 증가하는 값을 차례대로 대입

for (i = 0; i < m; i++)

for (j = 0; j < n; j++) {

setB[i][j] = value;

value += 2;

}

//행렬A+행렬B 출력

for (i = 0; i < m; i++) {

for (j = 0; j < n; j++)

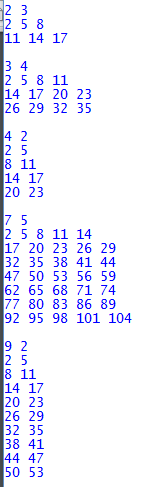
printf("%d ", setA[i][j] + setB[i][j]);

printf("\n");

}

}

**1.3. Result (snapshot)**



**2. 행렬의 곱을 출력**

**2.1. Solution**

행렬의 곱을 일단 보면 곱해지는 행렬의 행의 각각의 원소와 곱하려는 열의 각각의 원소를 곱하여 모두 더한 값이 결과 값의 원소가 된다. 따라서 결과는 항상 행렬의 행 M X M의 모습을 가지게 된다. 따라서 이러한 연산을 위해서는 3중 반복 문을 사용하면 된다. 행렬 A의 첫 번째 행의 원소들을 행렬 B의 열의 원소들과 곱하여 합한 값을 출력하는 방식으로 코드를 보면 변수 k를 추가하여 이를 가능케 하였다.

**2.2. Source code**

void arrayMul() // 행렬의 곱을 출력

{

int m, n; //m: 행, n: 열

int set1[MAX\_SIZE][MAX\_SIZE]; //행렬 A

int set2[MAX\_SIZE][MAX\_SIZE]; //행렬 B

int i, j, k;

int value = 1;

scanf("%d %d", &m, &n);

// m X n 행렬

for (i = 0; i < m; i++)

for (j = 0; j < n; j++)

set1[i][j] = value++;

value = 1;

// n X m 행렬

for (i = 0; i < n; i++) {

for (j = 0; j < m; j++)

set2[i][j] = value++;

}

value = 0;

k = 0;

// 행렬 A X 행렬 B 출력

for (i = 0; i < m; i++) {

for (j = 0; j < m; j++) {

while (k != n) {

//행렬 A의 각각의 행의 원소 X 행렬 B의 각각의 열의 원소

value += set1[i][k] \* set2[k][j];

k++;

}

//출력 및 변수 초기화

printf("%d ", value);

value = 0;

k = 0;

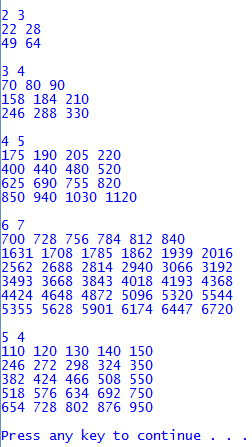
}

printf("\n");

}

}

**2.3. Result (snapshot)**



**3. 나선형 모양 행렬**

**3.1. Solution**

**3.2. Source code**

void spiralArray() // 나선형 모양 행렬

{

int m; //m: 열과 행

int i, j;

int set[MAX\_SIZE][MAX\_SIZE];

int value = 1;

int x, y; //x: 열 이동, y: 행 이동

int begin; //시작점

int end; //끝점

int repeat; //반복 횟수

scanf("%d", &m);

//행렬 생성

for (i = 0; i < m; i++)

for (j = 0; j < m; j++)

set[i][j] = value++;

//변수들 초기화

i = 0;

x = 0;

y = 0;

begin = 0;

end = m - 1;

//행의 길이 보다 1 작게 반복

repeat = m - 1;

while (1) {

//반복 횟수가 1이면 마지막 원소이기에 출력 후 종료

if (repeat == 1) {

printf("%d ", set[y][x]);

break;

}

for (i = 0; i < repeat; i++) //행렬의 좌상->우상

printf("%d ", set[begin][x++]);

for (i = 0; i < repeat; i++) //행렬의 우상->우하

printf("%d ", set[y++][end]);

for (i = 0; i < repeat; i++) //행렬의 우하->좌하

printf("%d ", set[end][x--]);

for (i = 0; i < repeat; i++) //행렬의 좌하->좌상

printf("%d ", set[y--][begin]);

//변수들 증가 및 감소, 초기화

y++;

x++;

begin++;

end--;

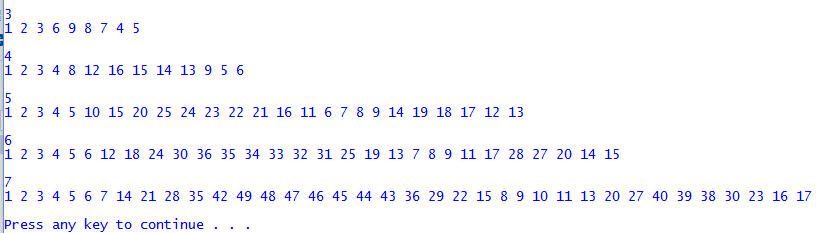
repeat /= 2;

}

printf("\n");

}

**3.3. Result (snapshot)**



**4. 가능한 Path의 개수**

**4.1. Solution**

**4.2. Source code**

**4.3. Result (snapshot)**

**5. Longest Path의 값 출력**

**5.1. Solution**

**5.2. Source code**

**5.3. Result (snapshot)**