(훈련반2) Level29

그래프 / 트리의 탐색의 두번째 방법인 BFS (Breadth Deapth Search, 너비우선탐색)을 시작해 봅시다.

DFS와 달리, **while과 큐**를 이용하여 구현합니다.

그래프를 탐색하는 두 가지 방법 DFS와 BFS 을 모두 익혀봅시다.

Level29 대문자만 DFS

문제 1번 [숙제 목록보기]

문자열로 구성된 이진트리를 입력 받아주세요.

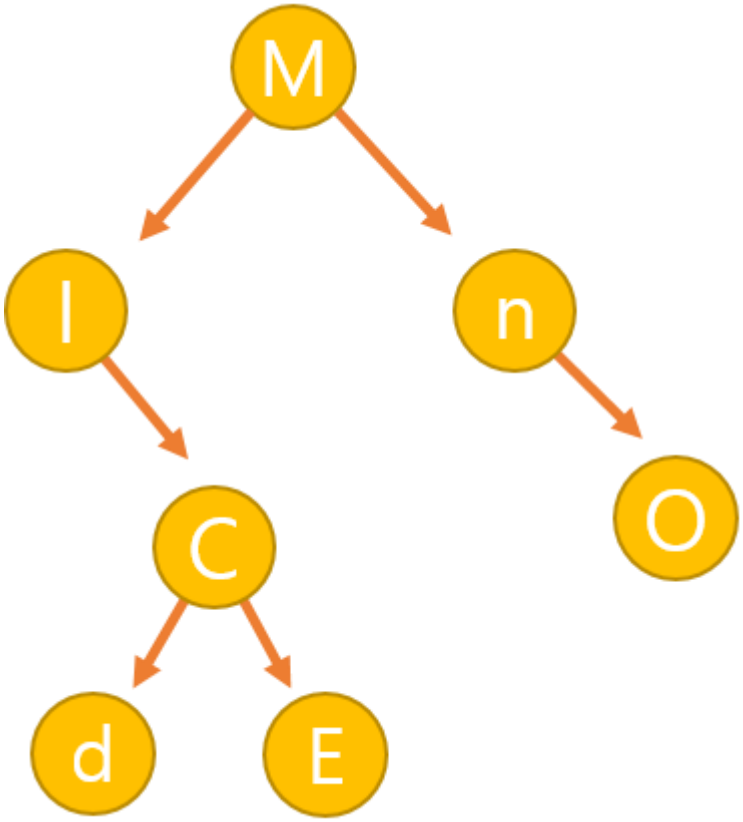
(1차원 배열에 저장하는 형태로 입력이 주어집니다.)

(#은 노드가 없음을 의미합니다.)

Root노드부터 DFS를 돌리면서

대문자 노드를 탐색할 때마다 출력 해 주세요.

만약 #MIn#C#O##dE을 입력받았다면



출력결과 : MICEO

입력 예제

```
#MIn#C#O##dE
```

출력 결과

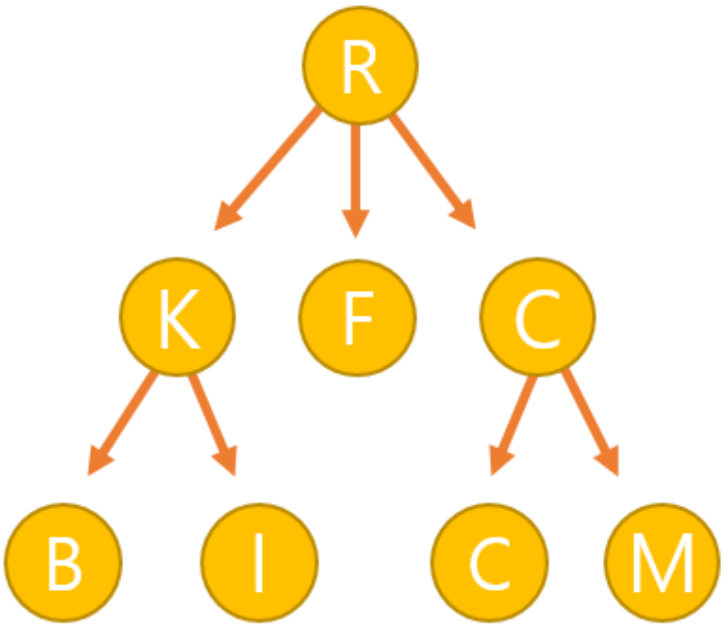
```
MICEO
```

```cpp
#include <iostream>
//#define _CRTDBG_MAP_ALLOC
//#include <stdlib.h>
//#include <crtdbg.h>
using namespace std;

/*
   입력값 : #MIn#C#O##dE
   - 이진트리 -> 일차원배열 : 해당 경우 배열의 0번 인덱스는 비워둔다.
   - 규칙 :
      1. Left : leftIdx = parentIdx * 2
      2. Right : rightIdx = parentIdx * 2 + 1

*/

char input[20] = {};
int inputSize = 0;
char* path = nullptr;
int pathLevel = 0;

void recursive(int _idx)
{
    char nodeName = input[_idx];
    if (_idx < inputSize)   // 입력값의 길이보다 작은지 확인
    {
        if (nodeName != '#') // '#' 은 노드가 없음을 의미한다.
        {
            path[pathLevel++] = nodeName;
            recursive(_idx * 2);        // left
            recursive(_idx * 2 + 1);   // right
        }
    }
}

int main(void)
{
    cin >> input;
    inputSize = strlen(input);          // 입력 값 길이
    path = new char[inputSize];         // 경로의 길이가 input 값의 길이를 넘을 순 없기에
inputSize로 메모리 할당
    for (int i = 0; i < inputSize; ++i) // 초기화
        path[i] = '\0';
    int startIdx = 1;                   // 초기 인덱스
    recursive(startIdx);

    int pathLen = strlen(path);
    // ascii -> 'A' = 65 , 'Z' = 90
```

```
for (int k = 0; k < pathLen; ++k)
    {
        if (path[k] >= 65 && path[k] <= 90)
            cout << path[k];
    }

    delete[] path;

    //_CrtDumpMemoryLeaks();
    return 0;
}
```

Level29 문자열 노드 DFS

문제 2번 [숙제 목록보기]
8개의 노드로 구성 된 문자열과
대응되는 인접행렬을 입력받아주세요.
아래 이미지는 입력 예시에 해당하는 트리입니다.
0번 노드부터 DFS로 노드들을 탐색하면서 출력 해 주세요.



입력 예제

```
RKFCBICM
0 1 1 1 0 0 0 0
0 0 0 0 1 1 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

출력 결과

```
RKBIFCCM
```

```cpp
#include <iostream>
using namespace std;
/*
        R K F C B I C M
        0 1 1 1 0 0 0 0  R
        0 0 0 0 1 1 0 0  K
        0 0 0 0 0 0 0 0  F
        0 0 0 0 0 0 1 1  C
        0 0 0 0 0 0 0 0  B
        0 0 0 0 0 0 0 0  I
        0 0 0 0 0 0 0 0  C
        0 0 0 0 0 0 0 0  M

*/

const bool yes = true;
const bool no = false;
const int nodeCnt = 8;                          // 노드이 개수
char nodes[nodeCnt + 1] = {};                   // 각 노드의 이름을 저장하는 배열
int matrix[nodeCnt][nodeCnt] = {};              // 노드간의 관계를 저장하는 인접행렬
char path[nodeCnt + 1] = {};                    // 방문하는 노드를 저장하는 배열(경로저장)
int pathIdx = 0;                                // 경로저장 배열에서 다음값이 들어갈 위치

void dfs(int _nodeIdx)
{
        path[pathIdx++] = nodes[_nodeIdx];      // 현재 노드의 이름을 경로에 넣어준다.
                                                // path배열에 다음값이 들어갈 위치를
                                                // 수정한다. (pathIdx++)
        for (int i = 0; i < nodeCnt; ++i)
        {
                int val = matrix[_nodeIdx][i];
                bool canIGo = (val == 1);
                if (canIGo == yes)
                {
                        dfs(i);
                }
        }
}

void getValue()                                 // 값을 입력받는다.
{
        cin >> nodes;
        for (int y = 0; y < nodeCnt; ++y)
        {
                for (int x = 0; x < nodeCnt; ++x)
                        cin >> matrix[y][x];
```

```
        }
}

void proc()                                     // 문제를 수행한다.
{
        int headNodeIdx = 0;                    // nodes[0] = 'R'
        dfs(headNodeIdx);
        cout << path;
}


int main(void)
{
        getValue();
        proc();

        return 0;
}
```

Level29 링크드리스트와 DFS

문제 3번 [숙제 목록보기]

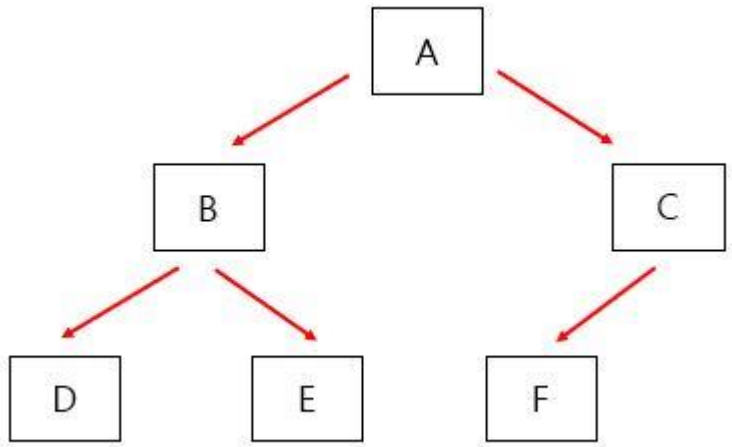
6글자를 입력받고 링크드리스트로 다음과 같이 연결시켜주세요.

ex)

```
head = new Node( );

head->left = new Node( );

head->right = new Node( );

...
```

링크드리스트 노드에 각각 입력받은 문자를 넣어주면 됩니다.

만약, ABCDEF 를 입력 받았다면 아래와 같습니다.



이제 DFS를 돌린결과를 출력 해주세요.

입력 예제

```
ABCDEF
```

출력 결과

```
ABDECF
```

```
#include <iostream>
#define _CRTDBG_MAP_ALLOC
#include <stdlib.h>
#include <crtdbg.h>
using namespace std;

const int nodeCnt = 6;
char nodes[nodeCnt + 1] = {};
char path[nodeCnt + 1] = {};
int pathIdx = 0;

class MyList
{
public:
	MyList()
		: head(nullptr)
	{

	}

	struct Node
	{
	public:
		Node()
			: name('\0')
			, left(nullptr)
			, right(nullptr)
		{
		}
		Node(char _name)
			: name(_name)
			, left(nullptr)
			, right(nullptr)
		{
		}
		char name;
		Node* left;
		Node* right;
	};

	Node* head;
};

MyList* list = nullptr;
```

```
void init()      // 초기 입력값 받고 값 세팅
{
        list = new MyList();
        cin >> nodes;
        list->head = new MyList::Node(nodes[0]);  // 형식지정자?
        list->head->left = new MyList::Node(nodes[1]);
        list->head->right = new MyList::Node(nodes[2]);
        list->head->left->left = new MyList::Node(nodes[3]);
        list->head->left->right = new MyList::Node(nodes[4]);
        list->head->right->left = new MyList::Node(nodes[5]);
}

void dfs(MyList::Node* _target)
{
        MyList::Node* target = _target;
        path[pathIdx++] = target->name;
        if (target->left != nullptr)
                dfs(target->left);
        if (target->right != nullptr)
                dfs(target->right);
}

void proc()      // 문제수행
{
        MyList::Node* firstNode = list->head;
        dfs(firstNode);
}

void printPath()
{
        cout << path;
}

void end()
{
        delete list->head->right->left;
        delete list->head->left->left;
        delete list->head->left->right;
        delete list->head->right;
        delete list->head->left;
        delete list->head;
        delete list;
}
```

```
int main(void)
{
        init();
        proc();
        printPath();
        end();

        _CrtDumpMemoryLeaks();
        return 0;
}
```

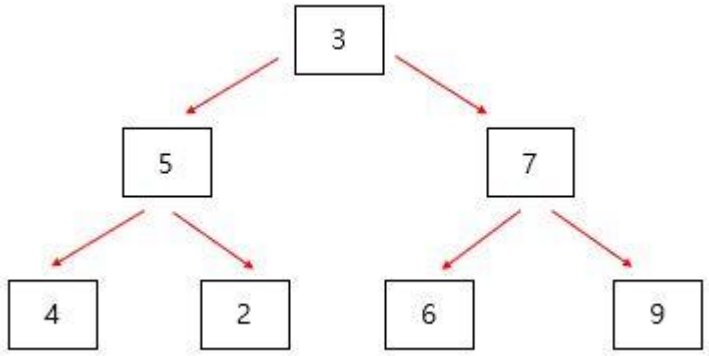
Level29 도착 경로의 합

문제 4번 [숙제 목록보기]

노드에 들어갈 숫자 7개를 입력 받으세요.

만약 **3 5 7 4 2 6 9** 를 입력 받았다면 다음과 같습니다.

(Root는 1번 index에 저장해야 함을 잊지마세요)



이제 **마지막 노드(Level2)에 있는 숫자들의 합을**

DFS를 돌려 구해주세요.

입력 예제

```
ㄴ
```

출력 결과

```
21
```

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

const int inputSize = 7;
int input[inputSize + 1] = {};
int levelTwoSum = 0;

struct Node
{
public:
	Node()
		: val(0)
		, level(0)
	{

	}
	int val;
	int level;
};

void init()
{
	for (int i = 1; i < inputSize + 1; ++i)
		cin >> input[i];
}

void dfs(int _idx, int _level)
{
	if (_level == 2)
		levelTwoSum += input[_idx];

	if (_idx < inputSize + 1)
	{
		int nextIdxL = _idx * 2;
		int nextIdxR = _idx * 2 + 1;
		int nextLevel = _level + 1;

		dfs(nextIdxL, nextLevel);
		dfs(nextIdxR, nextLevel);
	}

}
```

```
void proc()
{
	int startIdx = 1;
	int startLevel = 0;
	dfs(startIdx, startLevel);

	cout << levelTwoSum;
}

int main(void)
{
	init();
	proc();

	return 0;
}
```

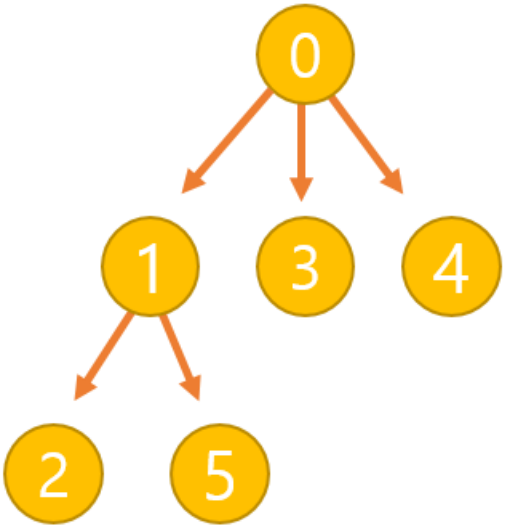
Level29 홀수만 BFS

문제 5번 [숙제 목록보기]

0 ~ 5번까지 6개 노드로 구성된 인접행렬을 입력받아주세요.

0번 노드부터 BFS를 돌려, **홀수 노드를 찾을 때 마다 출력** 해 주세요.

만약 위와 같이 트리를 입력받았다면,



출력결과 : 1 3 5

입력 예제

```
0 1 0 1 1 0
0 0 1 0 0 1
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

출력 결과

```
1 3 5
```

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

const int nodeCnt = 6;
int matrix[nodeCnt][nodeCnt] = {};
const int yes = 1;
const int no = 0;

struct Node
{
public:
	Node()
		: val(0)
		, level(0)
	{
	}
	int val;
	int level;
};

queue<Node> que;

void init()
{
	for (int y = 0; y < nodeCnt; ++y)
	{
		for (int x = 0; x < nodeCnt; ++x)
		{
			cin >> matrix[y][x];
		}
	}

	Node firstNode;
	firstNode.val = 0;
	firstNode.level = 0;
	que.push(firstNode);
}

void isOddNum(int num)
{
	if (num % 2 == 1)
		cout << num << " ";
}
```

```
void bfs()
{
	while (que.empty() == false)
	{
		Node target = que.front();
		for (int i = 0; i < nodeCnt; ++i)
		{
			int canIGo = matrix[target.val][i];
			if (canIGo == yes)
			{
				Node newNode;
				newNode.level = target.level + 1;
				newNode.val = i;
				que.push(newNode);
				isOddNum(newNode.val);
			}
		}
		que.pop();
	}
}

int main(void)
{

	init();
	bfs();

	return 0;
}
```

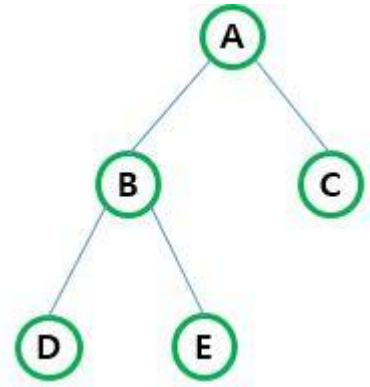
Level29 TREE DFS

문제 6번 [숙제 목록보기]

노드 안에 들어갈 문자 5개를 입력 받으세요.

만약 ABCDE 입력 받으면 아래와 같이 저장됩니다.

(트리모양은 고정입니다)



위 트리를 **인접행렬**로 저장하고, DFS를 돌려 탐색 순서대로 출력하세요.

(힌트 : 입력받은 문자열은 배열에 따로 저장해두어야 합니다)

입력 예제

```
ABCDE
```

출력 결과

```
ABDEC
```

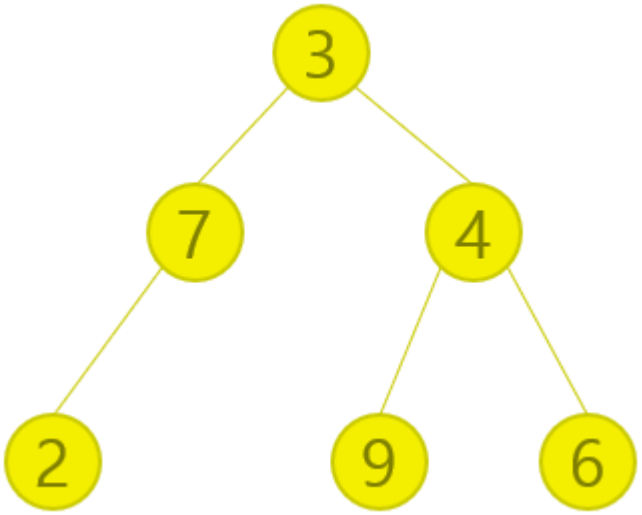
```cpp
#include <iostream>
using namespace std;

const int inputSize = 5;
char input[inputSize + 1] = {};
int isVisit[inputSize] = {};

int matrix[inputSize][inputSize] =
{
	//	a b c d e
		0,1,1,0,0,	// a
		1,0,0,1,1,	// b
		1,0,0,0,0,	// c
		0,1,0,0,0,	// d
		0,1,0,0,0,	// e
};

void dfs(int _idx)
{
	cout << input[_idx];
	isVisit[_idx] = 1;
	for (int i = 0; i < inputSize; ++i)
	{
		if (matrix[_idx][i] == 1 && isVisit[i] == 0)
		{
			dfs(i);
		}
	}
}

int main(void)
{
	cin >> input;
	int startIdx = 0;
	dfs(startIdx);
	return 0;
}
```

Level29 이진트리 BFS

문제 7번 [숙제 목록보기]

총 8개의 숫자를 1차원 배열에 입력받습니다. **(1~9 사이 숫자, 0은 없는 노드)**

BFS를 돌려 **탐색 순서대로 값을 출력** 해주세요.



만약 **0 3 7 4 2 0 9 6** 을 입력 받았다면 위와 같은 트리가 됩니다.

ex) **0 3 7 4 2 0 9 6** => **출력결과 : 3 7 4 2 9 6**

입력 예제

```
0 1 2 3 0 0 4 5
```

출력 결과

```
1 2 3 4 5
```

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

const int inputSize = 8;
int input[inputSize] = {};

struct Node
{
public:
	Node()
		: val(0)
		, matrixIdx(0)
		, level(0)
	{
	}
	int val;
	int matrixIdx;
	int level;
};
queue<Node> que;

int matrix[inputSize - 1][inputSize - 1] =
{
	//	1 2 3 4 5 6 7
		0,1,1,0,0,0,0,	// 1
		0,0,0,1,0,0,0,	// 2
		0,0,0,0,0,1,1,	// 3
		0,0,0,0,0,0,0,	// 4
		0,0,0,0,0,0,0,	// 5
		0,0,0,0,0,0,0,	// 6
		0,0,0,0,0,0,0,	// 7
};

void bfs()
{
	while (que.empty() == false)
	{
		Node target = que.front();
		for (int i = 0; i < inputSize-1; ++i)
		{
			if (matrix[target.matrixIdx][i] == 1 && input[i + 1] != 0)
			{
				Node newNode;
				newNode.val = input[i + 1];
				newNode.level = target.level + 1;
				newNode.matrixIdx = i;
```

```
                                que.push(newNode);
                        }
                }
                cout << que.front().val << " ";
                que.pop();
        }


}

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

        Node firstNode;
        firstNode.val = input[1];
        firstNode.level = 0;
        firstNode.matrixIdx = 0;
        que.push(firstNode);
        bfs();

        return 0;
}
```

Level29 터미네이터 신경망

문제 8번 [숙제 목록보기]

터미네이터의 신경체계가 다음과 같이 트리형태로 되어있다고 합니다.

인접행렬 형태로 초기화를 해 주세요.

이 신경체계를 트리로 하드코딩 하고, 나노 탐사로봇이 탐색을 하려고 합니다.

BFS 알고리즘을 쓸때 탐사순서대로 출력해주세요.

예를들어 E를 입력받았다면,

E에서 출발하는 BFS를 돌리면 됩니다. (출력결과 : E H J)

예를들어 B를 입력받았다면,

B에서 출발하는 BFS를 돌리면 됩니다. (출력결과 : B C D E F G H I J)

입력 예제

```
A
```

출력 결과

```
A B C D E F G H I J
```

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

const int nodeCnt = 10;
char nodeNames[nodeCnt + 1] = "ABCDEFGHIJ";
int matrix[nodeCnt][nodeCnt] =
{
//      a b c d e f g h i j
        0,1,0,0,0,0,0,0,0,0, // a
        0,0,1,1,1,1,0,0,0,0, // b
        0,0,0,0,0,0,0,0,0,0, // c
        0,0,0,0,0,0,1,0,0,0, // d
        0,0,0,0,0,0,0,1,0,0, // e
        0,0,0,0,0,0,0,0,0,0, // f
        0,0,0,0,0,0,0,0,1,0, // g
        0,0,0,0,0,0,0,0,0,1, // h
        0,0,0,0,0,0,0,0,0,0, // i
        0,0,0,0,0,0,0,0,0,0, // j
};

struct Node
{
public:
        Node()
                : name('\0')
                , idx(0)
                , level(0)
        {

        }
        char name;
        char idx;
        int level;
};

queue<Node> que;

void bfs()
{
        while(que.empty() == false)
        {
                Node target = que.front();
                for (int i = 0; i < nodeCnt; ++i)
                {
                        if (matrix[target.idx][i] == 1)
                        {
```

```
                                Node newNode;
                                newNode.name = nodeNames[i];
                                newNode.idx = i;
                                newNode.level = target.level + 1;
                                que.push(newNode);
                        }
                }
                cout << que.front().name << " ";
                que.pop();
        }

}

int main(void)
{
        char startNodeName = '\0';
        cin >> startNodeName;
        int startNodeIdx = 0;
        for (int i = 0; i < nodeCnt; ++i)
        {
                if (nodeNames[i] == startNodeName)
                {
                        startNodeIdx = i;
                        break;
                }
        }
        Node startNode;
        startNode.name = nodeNames[startNodeIdx];
        startNode.idx = startNodeIdx;
        startNode.level = 0;
        que.push(startNode);
        bfs();

        return 0;
}
```

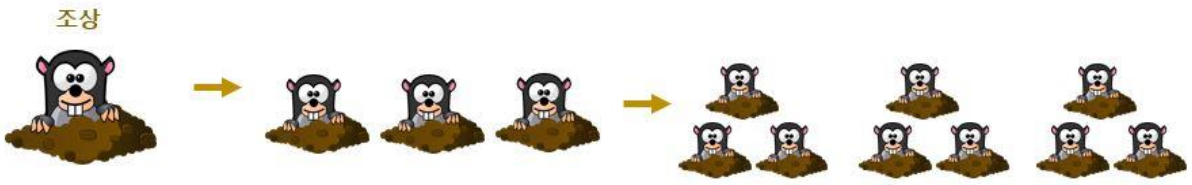
Level29 번식왕 두더지

문제 9번 [숙제 목록보기]

조상 두더지는 1년에 세마리의 자식을 낳습니다.

1년 후 각각의 자식들은 각자 세마리의 자식을 낳습니다.

큐를 이용해서 n년 후에는 총 몇마리의 두더지가 있는지 출력 해주세요.



ex)

0년 => 1마리

1년 => 3마리 + 1마리 = 4마리

2년 => 9마리 + 3마리 + 1마리 = 13마리

[힌트]

Queue에 1을 넣어두고 시작합니다.

Head index의 값에 * 3 곱한 값을 큐에 추가하고, Head++ (큐에 3이 적혀집니다)

Head index의 값에 * 3 곱한 값을 큐에 추가하고, Head++ (큐에 9가 적혀집니다)

이 과정을 n번 반복한 후

큐에 적혀있는 숫자들을 모두 더하면 됩니다.

입력 예제

```
1
```

출력 결과

```
4
```

```cpp
#include <iostream>
#include <queue>
using namespace std;
const int babyCnt = 3;
int inputYear = 0;
struct Node
{
public:
	Node()
		: amount(1)
		, year(0)
	{

	}
	int amount;
	int year;
};

queue<Node> que;
int year = 0;
int totalMoleCnt = 0;

void bfs()
{
	while (que.empty() == false)
	{
		Node target = que.front();
		if (target.year > inputYear)
		{
			break;
		}
		for (int i = 0; i < babyCnt; ++i)
		{
			Node newMole;
			newMole.year = target.year + 1;
			que.push(newMole);
		}
		totalMoleCnt += que.front().amount;
		que.pop();
	}

}
```

```
int main(void)
{
        cin >> inputYear;
        Node fisrtMole;  // mole : 두더지
        que.push(fisrtMole);
        bfs();
        cout << totalMoleCnt;
        return 0;
}
```