```
#include<stdio.h>
#define MAX_NODE 100000
#define UNDEFINED (-1)
//-----------------------------------------------------------------------------------------------
------------------------------------
/* linked list info */
struct NODE
{
        int id;
        struct NODE * prev;
        struct NODE * next;
        struct NODE * last;
}array[MAX_NODE+1];
int arrIdx;

NODE * myalloc( )
{
        return &array[arrIdx++];
}

//-----------------------------------------------------------------------------------------------
------------------------------------

/* Tree info */
int parent_info[MAX_NODE+1];   //부모 정보를 저장할 배열
NODE *child_info[MAX_NODE+1];  //child 정보를 저장할 linked list

char name[MAX_NODE+1][32];
int age[MAX_NODE+1];
int NumberOfChild[MAX_NODE+1];

//-----------------------------------------------------------------------------------------------
------------------------------------
/* Queue Info */
int QUE[MAX_NODE+1];
int frontQ,endQ;

void enQ(int id)
{
        QUE[endQ++] = id;
}

int deQ()
{
        if( frontQ == endQ)
                return -1000;
        else
                return QUE[frontQ++];
}
//-----------------------------------------------------------------------------------------------
------------------------------------

void addTreeInfo(int pIdx, int cIdx);
void local_init();
int  getRootIdx();
void DFSSearch(int parent);
void BFSSearch(int parent);

int main(void) {

        local_init();

        addTreeInfo(7,3);
        addTreeInfo(7,4);
        addTreeInfo(7,16);
        addTreeInfo(7,5);
        addTreeInfo(3,8);
        addTreeInfo(3,17);
        addTreeInfo(3,1);
        addTreeInfo(4,0);
        addTreeInfo(4,14);
        addTreeInfo(5,2);
        addTreeInfo(5,26);
        addTreeInfo(17,6);

        //printf(" Root is %d ₩n",getRootIdx());
        DFSSearch(7); printf("₩n");
        BFSSearch(7);
}


void addTreeInfo(int pIdx, int cIdx)
{
        parent_info[cIdx] = pIdx;  //부모 정보를 child에 저장

        NODE * pNew = myalloc();
        pNew->id = cIdx;
        pNew->next=0;
        pNew->last=0;

        if( child_info[pIdx] == 0)
        {
                child_info[pIdx] = myalloc();
                pNew->prev = child_info[pIdx];
                child_info[pIdx]->next = pNew;
                child_info[pIdx]->last = pNew;
        }
        else
        {
                pNew->prev = child_info[pIdx]->last;
                child_info[pIdx]->last->next = pNew;
                child_info[pIdx]->last = pNew;   //주의 누락하지 말자!
        }

}

void local_init()
{
        arrIdx=0;
        for(int i=0 ; i<=MAX_NODE; i++ )
        {
                parent_info[i]=UNDEFINED;
                child_info[i]=0;
        }

        frontQ=0,endQ=0;
}

int getRootIdx()
{
        int child=0, parent=0;
        for(int i=MAX_NODE ; i>=0 ; i-- )
        {
                if( parent_info[i] != UNDEFINED)
                {
                        child = i;
                        parent = parent_info[child];
                        break;
                }
        }
        //printf(" %d %d ₩n",child,parent);
        while(parent!=UNDEFINED)
        {
                child = parent;
                parent = parent_info[child];
        }
```

```
	return child;
}


void DFSSearch(int parent)
{
	printf("DFS I'm [%d] ₩n",parent);
	//주의 null 체크 필요!
	if( child_info[parent] ==0)
		return;

	NODE *pChilds = child_info[parent]->next;
	int id;
	while(pChilds)
	{
		id = pChilds->id;
		DFSSearch(id);
		pChilds = pChilds->next;
	}
	return;
}


void BFSSearch(int parent)
{
	enQ(parent);
	while(1)
	{
		parent = deQ();
		if( parent == -1000)
		{
			break;
		}

		printf("BFS I'm [%d] ₩n",parent);
		//주의 null 체크 필요!
		if( child_info[parent] ==0)
			continue;	//주의 continue 로 흐름 제어!

		NODE *pChilds = child_info[parent]->next;
		int id;
		while(pChilds)
		{
			id = pChilds->id;
			enQ(id);
			pChilds = pChilds->next;
		}
	}

}
```