/\*\*

 \* Definition for a binary tree node.

 \* struct TreeNode {

 \*     int val;

 \*     struct TreeNode \*left;

 \*     struct TreeNode \*right;

 \* };

 \*/

/\*\*

 \* Return an array of arrays of size \*returnSize.

 \* The sizes of the arrays are returned as \*returnColumnSizes array.

 \* Note: Both returned array and \*columnSizes array must be malloced, assume caller calls free().

 \*/

void putin(int val, int level , int\* returnSize, int\*\* returnColumnSizes, int \*\*result){

  if( \*returnSize < level ){//如果現在的level與最大的\*returnSize比較，level比較大

    \*returnColumnSizes=realloc(\*returnColumnSizes,sizeof(int)\*(\* returnSize+1 ));//這樣就增加\*returnColumnSizes的空間，紀錄ColumnSizes

    result[\* returnSize]=malloc(sizeof(int\*)\*1);//並且再result[\* returnSize]增加一個空間給val

    //printf("%d\n",val);

    result[\* returnSize][0]=val;

    (\*returnColumnSizes)[\*returnSize]=1;//將\*returnColumnSizes的空間放入1初始化

    (\*returnSize)++;

  }else{//如果level小於\*returnSize表示已應有建立過這個level在result中了

    (\*returnColumnSizes)[level-1]++;//將\*returnColumnSizes的空間中的計數++

    result[level-1]=realloc(result[level-1],sizeof(int)\* ((\*returnColumnSizes)[level-1]) );//將result[level]的空間再增加

    result[level-1][ (\*returnColumnSizes)[level-1]-1 ]=val;//把值放到新增加的空間中

  }

}

void run (struct TreeNode\* root,int level, int\* returnSize, int\*\* returnColumnSizes, int \*\*result){

  //走完整個root

  if(root != NULL){

    (level)++;//level計算該節點是level多少

    //printf("%d level=%d\n",root->val,level);

    putin(root->val, level,returnSize,returnColumnSizes,result);//將找到的節點資料val與level塞進答案result中

  }else{

    return;

    }

  run ( root -> left,level,returnSize,returnColumnSizes,result);

  run ( root -> right,level,returnSize,returnColumnSizes,result);

}

int\*\* levelOrder(struct TreeNode\* root, int\* returnSize, int\*\* returnColumnSizes){

  int \*\*result=NULL;

  result=realloc(result,sizeof(int\*)\*(2000 +1));//宣告result[0][]~result[2000][]個空間

  int level=0;

  \* returnSize=0;//計算有多少個result[0][]~result[returnSize][]

  \*returnColumnSizes=malloc(sizeof(int)\*1);//returnColumnSizes[\*returnSize][x]中計算有多少x

  run(root,level,returnSize,returnColumnSizes,result);

  //for(int i=0;i<\*returnSize;i++){

  //   printf("returnColumnSizes[%d]=%d\n",i,(\*returnColumnSizes)[i]);

  // }

  // int \*\*ans=malloc(sizeof(int\*)\*1);

  // ans=realloc(ans,sizeof(int\*)\*1);

  // ans[0]=malloc(sizeof(int));

  // ans[0][0]=3;

  // ans=realloc(ans,sizeof(int\*)\*2);

  // ans[1]=malloc(sizeof(int)\*1);

  // ans[1][0]=9;

  // ans[1]=realloc(ans[1],sizeof(int)\*2);

  // ans[1][1]=20;

  // \* returnSize=2;

  // \*returnColumnSizes=realloc(\*returnColumnSizes,sizeof(int)\*2);

  // (\*returnColumnSizes)[0]=1;

  // (\*returnColumnSizes)[1]=2;

return result;

}