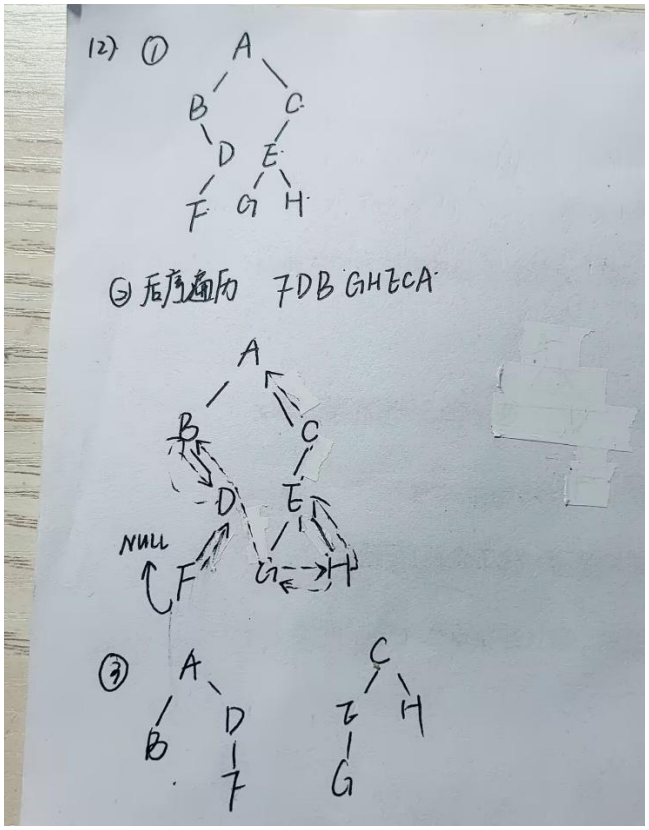
（2）



（5）

```
#include <stdio.h>
#include <stdlib.h>

// 二叉树节点结构
typedef struct TreeNode {
    int data;
    struct TreeNode* left;
    struct TreeNode* right;
} TreeNode;

// 队列节点结构（用于层次遍历）
typedef struct QueueNode {
    TreeNode* treeNode;
    struct QueueNode* next;
} QueueNode;

// 队列结构
typedef struct Queue {
    QueueNode* front;
    QueueNode* rear;
} Queue;
```

```
// 创建队列
Queue* createQueue() {
    Queue* queue = (Queue*)malloc(sizeof(Queue));
    queue->front = queue->rear = NULL;
    return queue;
}

// 入队操作
void enqueue(Queue* queue, TreeNode* treeNode) {
    QueueNode* newNode = (QueueNode*)malloc(sizeof(QueueNode));
    newNode->treeNode = treeNode;
    newNode->next = NULL;

    if (queue->rear == NULL) {
        queue->front = queue->rear = newNode;
    } else {
        queue->rear->next = newNode;
        queue->rear = newNode;
    }
}

// 出队操作
TreeNode* dequeue(Queue* queue) {
    if (queue->front == NULL) {
        return NULL;
    }

    QueueNode* temp = queue->front;
    TreeNode* treeNode = temp->treeNode;
    queue->front = queue->front->next;

    if (queue->front == NULL) {
        queue->rear = NULL;
    }

    free(temp);
    return treeNode;
}

// 判断队列是否为空
int isQueueEmpty(Queue* queue) {
    return queue->front == NULL;
}
```

```
// 计算二叉树的最大宽度
int maxWidth(TreeNode* root) {
    if (root == NULL) {
        return 0;
    }

    Queue* queue = createQueue();
    enqueue(queue, root);
    int maxWidth = 0;

    while (!isQueueEmpty(queue)) {
        int levelSize = 0;
        QueueNode* current = queue->front;
        QueueNode* levelEnd = queue->rear;

        // 计算当前层的节点数
        while (1) {
            levelSize++;
            if (current == levelEnd) break;
            current = current->next;
        }

        // 更新最大宽度
        if (levelSize > maxWidth) {
            maxWidth = levelSize;
        }

        // 处理当前层的所有节点，并将下一层节点入队
        int count = levelSize;
        while (count > 0) {
            TreeNode* currentNode = dequeue(queue);

            if (currentNode->left != NULL) {
                enqueue(queue, currentNode->left);
            }
            if (currentNode->right != NULL) {
                enqueue(queue, currentNode->right);
            }
            count--;
        }
    }

    free(queue);
    return maxWidth;
```

```
}

// 创建新节点
TreeNode* createNode(int data) {
    TreeNode* newNode = (TreeNode*)malloc(sizeof(TreeNode));
    newNode->data = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

// 测试代码
int main() {
    // 创建测试二叉树:
    //          1
    //        / \
    //       2   3
    //      / \   \
    //     4   5   6
    // 最大宽度为 3（第 3 层有 3 个节点）

    TreeNode* root = createNode(1);
    root->left = createNode(2);
    root->right = createNode(3);
    root->left->left = createNode(4);
    root->left->right = createNode(5);
    root->right->right = createNode(6);

    int width = maxWidth(root);
    printf("二叉树的最大宽度为: %d\n", width);

    return 0;
}
```