**考试时间：15：00~17：30**

**要求：粘贴源代码+（需要结果的编程题）运行结果截图**

**（交给黄队）**

**1.设有一个二维数组A[m][n]，假设A[0][0]存放位置在644，A[2][2]存放位置在676，每个元素占一个空间，问A[3][3]存放在什么位置（C）(题中所有数值均为十进制)**

**A. 688 B. 678 C. 692 D. 696**

**2.设顺序循环队列Q[0:M-1]的头指针和尾指针分别为F和R，头指针总是指向队头元素的前一位置，尾指针总是指向队尾元素的当前位置，则该循环队列中元素的个数为（C）**

**A. R-F B. F-R C. (R-F+M)%M D. (F-R+M)%M**

**3. 如果结点A有 3个兄弟，而且B是A的双亲，则B的度是（B）**

**A. 3 B. 4 C. 5 D. 6**

**4.已知一棵二叉树的前序遍历结果为ABDGHCEIKFJ，中序遍历结果为GDHBAEKICFJ，则后序遍历的结果应为（B）**

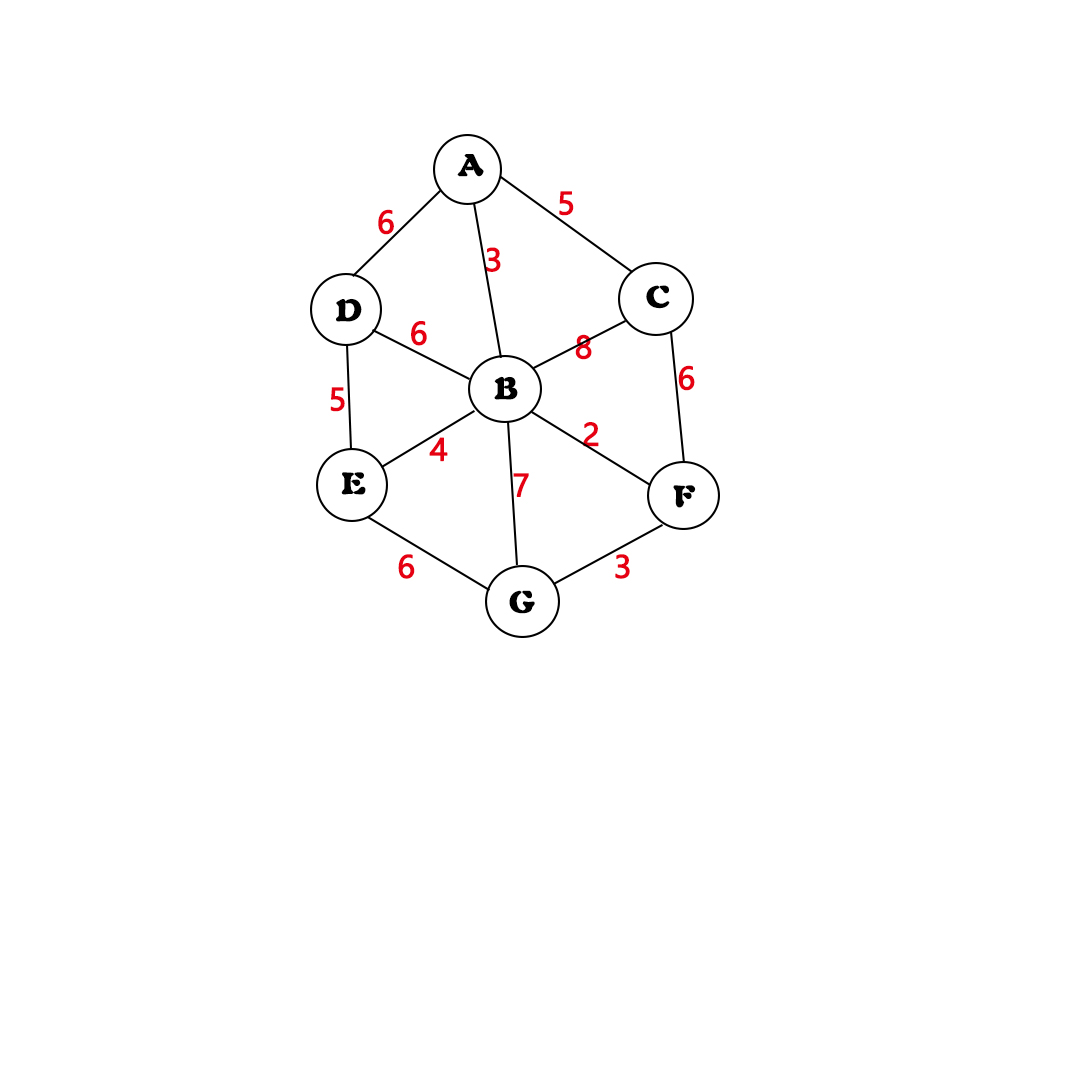
**A. GHDBCKIEJFA**

**B. GHDBKIEJFCA**

**C. BCADEFGHIJK**

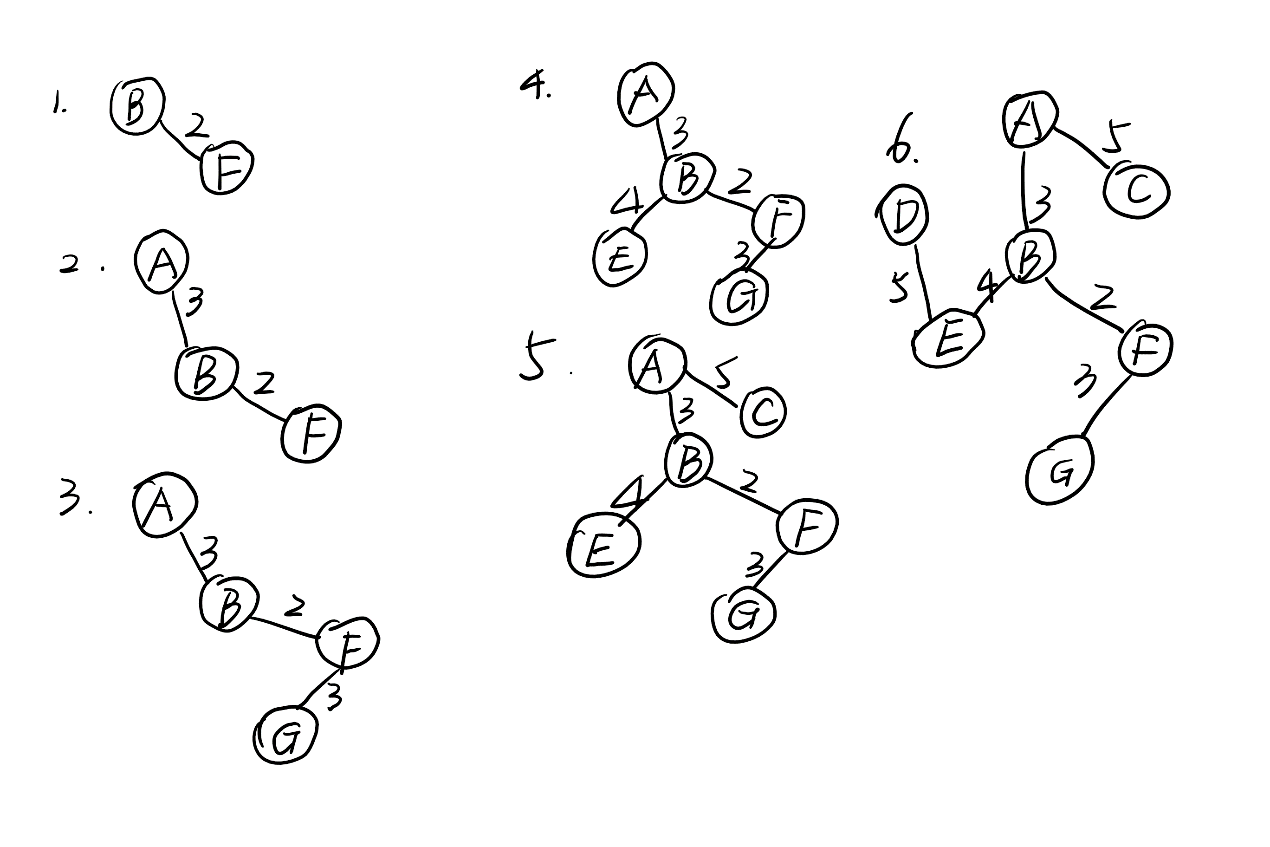
**D. BCAGHDKIJEF**

5.

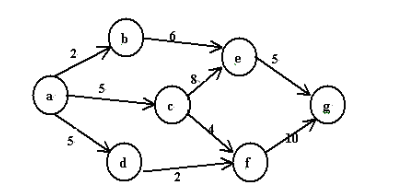


**求上述无向图的最小生成树，要求：只需画出每一步完成后当前已构成的树**

**(结果手写，拍照粘贴)**

****

**6.对于下图计算各个活动的最早发生时间和最迟发生时间，并求出该图中的关键路径**



|  |  |  |
| --- | --- | --- |
| **顶点** | **最早发生时间** | **最迟发生时间** |
| **A** | **0** | **0** |
| **B** | **2** | **8** |
| **C** | **5** | **5** |
| **D** | **5** | **7** |
| **E** | **13** | **14** |
| **F** | **9** | **9** |
| **G** | **19** | **19** |

|  |  |  |
| --- | --- | --- |
| **边** | **最早开始时间** | **最晚开始时间** |
| **<a, b>** | **0** | **6** |
| **<a, c>** | **0** | **0** |
| **<a, d>** | **0** | **2** |
| **<b, e>** | **2** | **8** |
| **<c, e>** | **5** | **6** |
| **<c, f>** | **5** | **5** |
| **<d, f>** | **5** | **7** |
| **<e, g>** | **13** | **14** |
| **<f, g>** | **9** | **9** |

**关键路径<a,c>,<c,f>,<f,g>**

**7.给定一个链表和一个特定值 x，对链表进行分隔，使得所有小于 x 的节点都在大于或等于 x 的节点之前，你应当保留两个分区中每个节点的初始相对位置。**

**示例:**

**输入: head = 1->4->3->2->5->2, x = 3**

**输出: 1->2->2->4->3->5 //分区1 2 2 和 4 3 5 保持原始相对位置**

**说明：**

**你不能只是单纯的改变节点内部的值，而是需要实际的进行节点交换。**

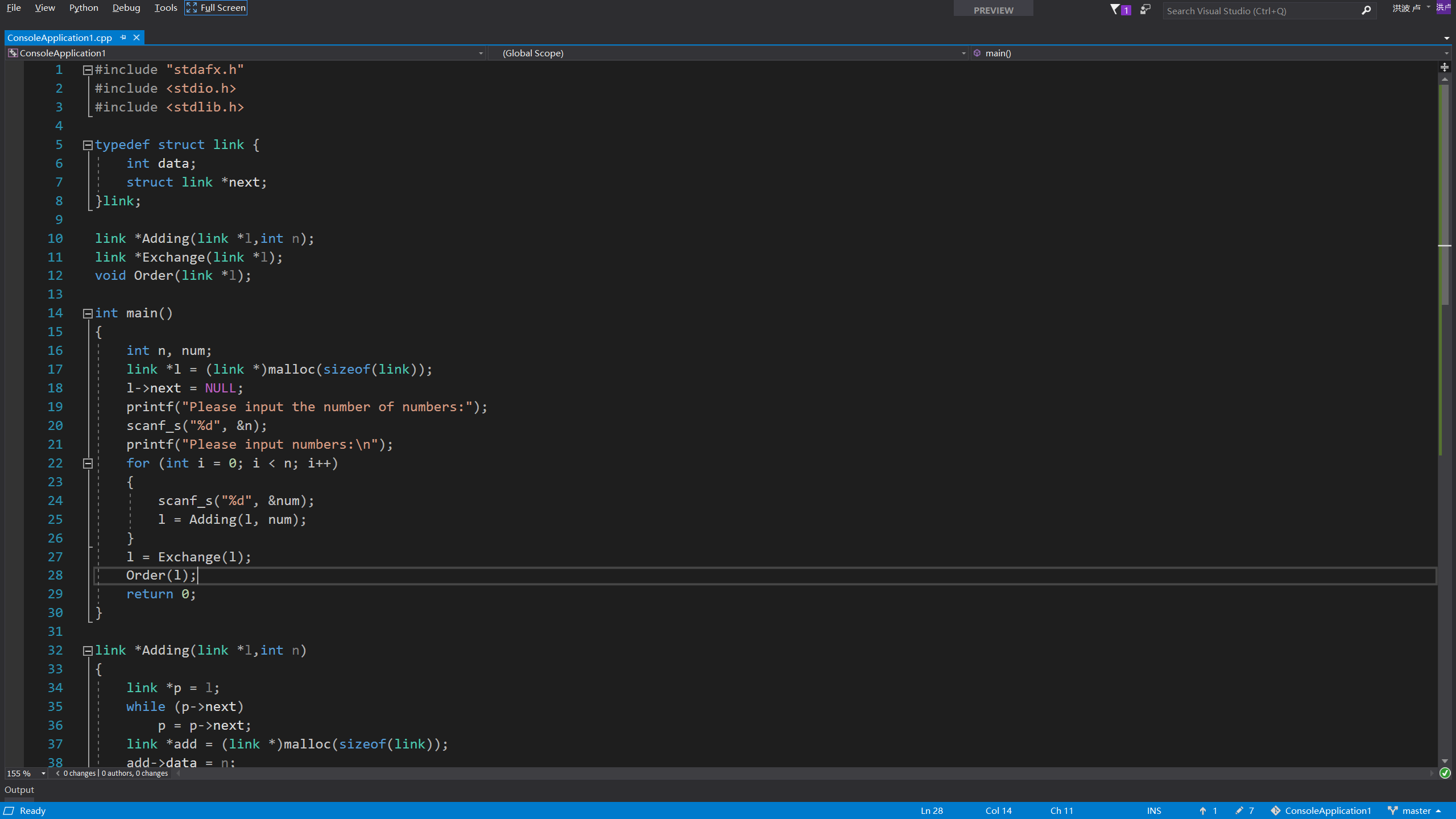
**//结构体如下**

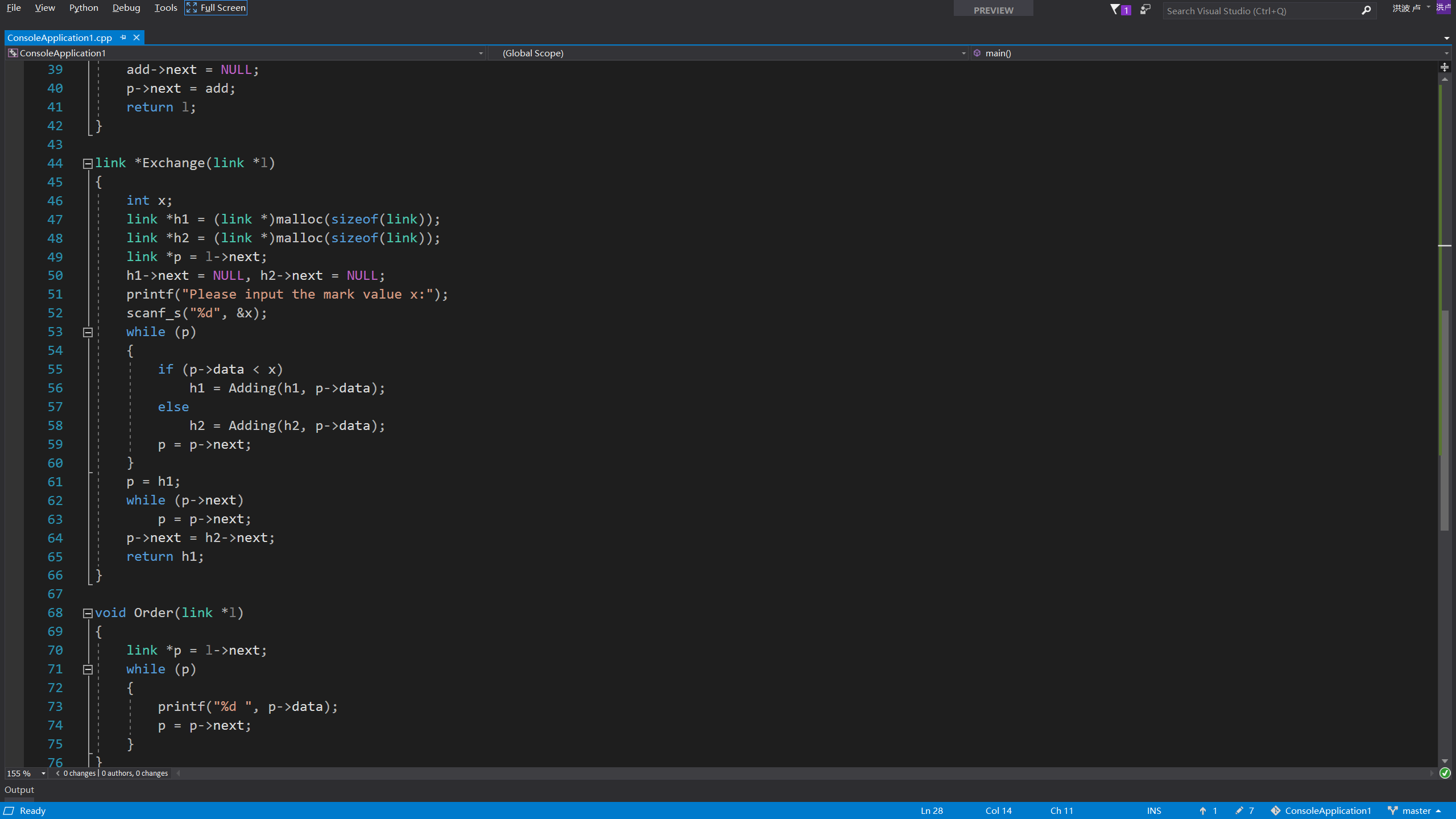
**typedef struct List\_node {**

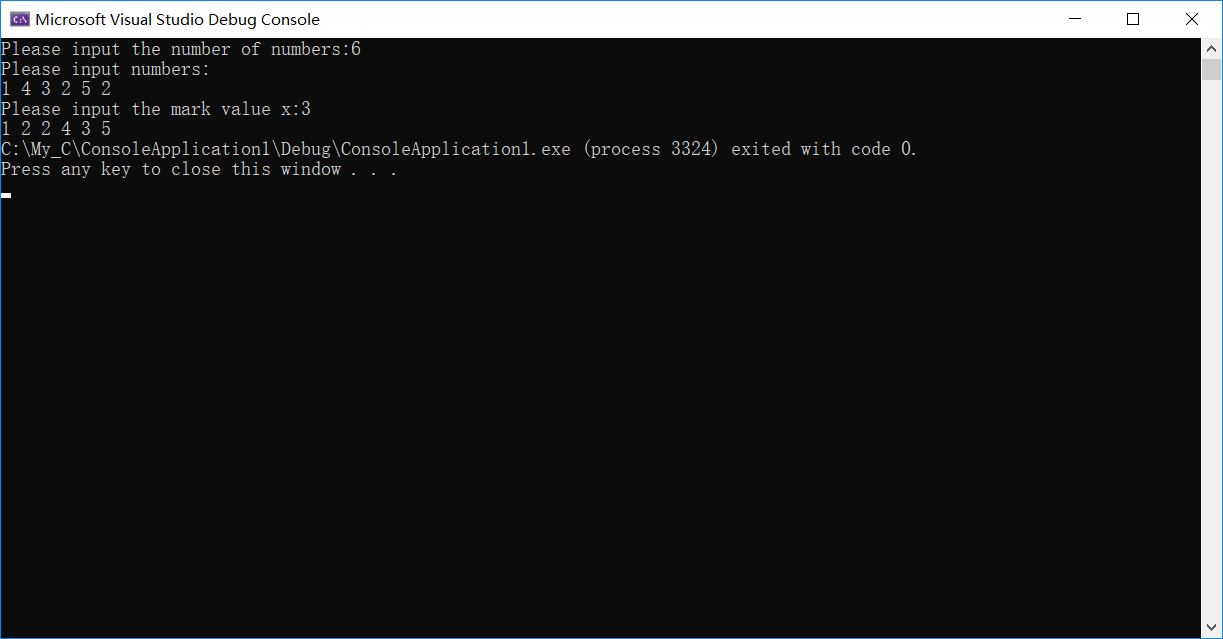
**int val;**

**struct List\_node\* next;**

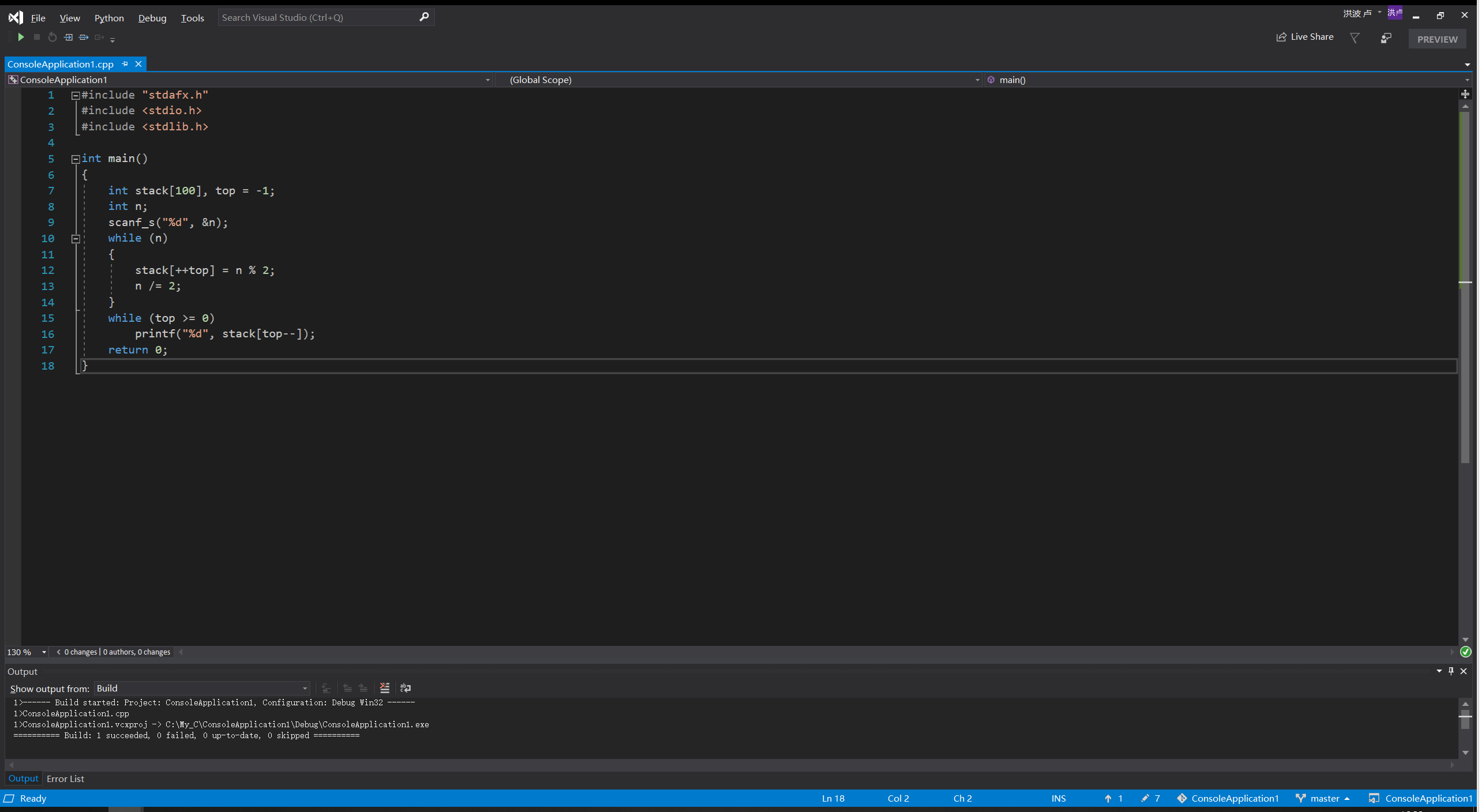
**} ListNode;**

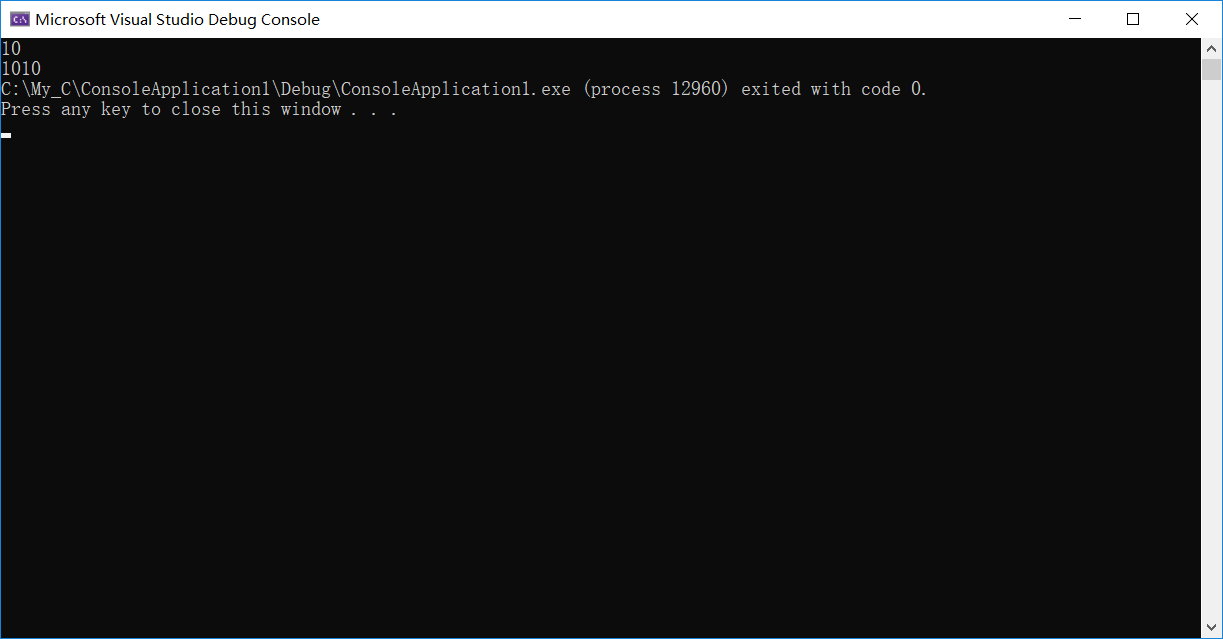






**8.用户提供了一个十进制数：10，要求将此数据以二进制形式转换，则通过进制转换器转换的最终结果应该：1010**





**9.将二叉树BT中每一个结点的左右子树互换的C语言算法如下，其中ADDQ(P,BT),DELQ(P),EMPTY(P)分别为进队，出队和判别队列是否为空的函数，请填写算法中得空白处，完成其功能。**

**typedef struct node**

**{int data ; struct node \*lchild, \*rchild; }btnode;**

**void EXCHANGE(btnode \*BT)**

**{btnode \*p, \*q;**

**if (BT){ADDQ(P,BT);**

**while(!EMPTY(P))**

**{p=DELQ(P); q=(1)\_ p->rchild \_\_;**

**p->rchild=(2)\_ p->lchild \_\_;**

**(3)\_\_p->lchild \_=q;**

**if(p->lchild) (4)\_ EXCHANGE(bt p->lchild) \_\_;**

**if(p->rchild) (5)\_ EXCHANGE(bt p->rchild) \_\_;**

**}**

**} }**

**10.以下是克鲁斯卡尔算法的一部分**

**//为边集数组Edge排序**

**qsort(edges, G.numEdges, sizeof(Edge), cmp);**

**for (i = 0; i < G.numVertexes; i++)**

**parent[i] = 0;**

**printf("打印最小生成树：\n");**

**for (i = 0; i < G.numEdges; i++) {**

**n = Find(parent, edges[i].begin);//寻找边edge[i]的“首节点”所在树的树根**

**m = Find(parent, edges[i].end);//寻找边edge[i]的“尾节点”所在树的树根**

**//假如n与m不等，说明两个顶点不在一棵树内，因此这条边的加入不会使已经选择的边集产生回路**

**if (n != m) {**

**parent[n] = m;**

**printf("(%d, %d) %d\n", edges[i].begin, edges[i].end, edges[i].weight);**

**}**

**}**

**写出Find()函数的具体代码 int Find(int \*parent, int f)**

{

while(parent[f])

f=parent[f];

return f;

{

**附加题：**

**若有一个字符集的字符个数为4个，每个字符出现的概率为37%、25%、18%、20%，请编写代码，运行程序输出每个字符的赫夫曼编码**

