阅读下列说明和C函数代码，将应填入（n）处的字句写在答题纸的对应栏内。（15分）

【说明】

对二叉树进行遍历是二叉树的一个基本运算。遍历是指按某种策略访问二叉树的每个结点，且每个结点仅访问一次的过程。函数**InOrder()借助栈实现二叉树的非递归中序遍历运算。**

设二叉树采用二叉链表存储，结点类型定义如下：

1. **typedef** **struct** **BtNode**{
2. ElemType    data;
3. /\*结点的数据域，ElemType 的具体定义省略\*/
4. **struct** BtNode \*lchild,\*rchild;
5. /\*结点的左、右孩子指针域\*/
6. }BtNode, **\*BTree**;

在函数 InOrder()中，用栈暂存二叉树中各个结点的指针，并将栈表示为不含头结点的单向链表（简称链栈），其结点类型定义如下：

**typedef** **struct** StNode{ /\*链栈的结点类型\*/

BTree elem;  /\*栈中的元素是指向二叉链表结点的指针\*/

**struct** StNode \*link;

}StNode;

假设从栈顶到栈底的元素为en、en-1、…、e1，则不含头结点的链栈示意图如图9-11所示。



【C函数】

int InOrder(BTree root) /\* 实现二叉树的非递归中序遍历 \*/

{

BTree ptr;/\* ptr用于指向二叉树中的结点 \*/

StNode \*q;  /\* q 暂存链栈中新创建或待删除的结点指针\*/

StNode \*stacktop = NULL; /\* 初始化空栈的栈顶指针stacktop \*/

ptr = root; /\* ptr指向二叉树的根结点 \*/

while (   （1）   || stacktop != NULL) {

while (ptr != NULL) {

q = (StNode \*)malloc(sizeof(StNode));

if (q == NULL)

return -1;

q->elem = ptr;

  （2）  ;

stacktop = q;  /\* stacktop指向新的栈顶 \*/

ptr =     （3）  ;  /\* 进入左子树 \*/

}

q = stacktop;

  （4）  ;     /\* 栈顶元素出栈 \*/

visit(q);  /\* visit是访问结点的函数，其具体定义省略 \*/

ptr =     （5）  ;   /\* 进入右子树 \*/

free(q);   /\* 释放原栈顶元素的结点空间 \*/

}

return 0;

}/\*InOrder\*/

答案：（1）ptr！=NULL （2）q->link=stacktop

(3) ptr=ptr->lchild (4)stacktop=stacktop->link (5)q->rchild