代码中缺省的部分：

struct intList{

int data;

struct intList\* next;

};

typedef struct intList\* IntList;

缺省了PVoidList, BpNode的结构声明

BpNode结构中有：

1.int flag或者是bool flag，应该是判断节点是否为叶子节点，如果是叶子节点那么flag=1，不是叶子节点flag=1。

bool isroot

2.Node\_P[ ]这个数组应该调用的另一个类型，有C->Node\_P[i + 1]->vaild != 0出现，那么valid应该是bool类型

C = C->Node\_P[14]->next\_node，其中next\_node是BpNode类型

3.还有一个K[ ]的数组

4.还有File\_P[i]，哇这又是什么L->File\_P[j]->lines = L->File\_P[j - 1]->lines

还有一个voidList结构体

(BNpointer)malloc(sizeof(struct Bplus\_Node\_Pointer)

(BFpointer)malloc(sizeof(struct Bplus\_File\_Pointer))这两个应该是把前面的结构体重新命名了

void BpBuild(int IID, unsigned char type, unsigned char size, IntList lines, PVoidList values);

建立B+树，其中lines和values的数据一一对应，表示<指针（行号），数据值>

将此树存为index6.bin

void BpDestroy(BpNode,int IID, unsigned char type, unsigned char size);

写入bin文件后，回收此B+树，防止占用存储空间。若建树过程中不使用malloc，此函数可以不写

void BpInsert(int IID, unsigned char type, unsigned char size, int line, void\* value);

把index6.bin读为一棵B+树，插入一个<指针（行号），数据值>，再存回index6.bin

IntList BpLess(int IID, unsigned char type, unsigned char size, void\* value, IntList il);

将index6.bin读出的B+树中，所有数据值小于value的对应的行号存入整数链表il中，返回il。

以下函数均类似，除了BpEqual只需要返回一个行号即可。

IntList BpLessEqual(int IID, unsigned char type, unsigned char size, void\* value, IntList il);

小于等于

int BpEqual(int IID, unsigned char type, unsigned char size, void\* value);

等于

IntList BpMoreEqual(int IID, unsigned char type, unsigned char size, void\* value, IntList il);

大于等于

IntList BpMore(int IID, unsigned char type, unsigned char size, void\* value, IntList il);

大于

IntList insert\_il(IntList List, int X)

向IntList中插入数据，插在链表头的位置，返回整个链表

PVoidList insert\_vl(PVoidList List, void\* X)

同上，不同的是，这里的链表是一个异质链表

void print(IntList List)

打印链表

BpNode find(BpNode Bpt, unsigned char type, void\* V, BpNode parent[])

在B+树中，找到Bpt这个节点并返回

输入Bpt节点，type是这个节点数据的类型，传入V的地址

返回节点C

BpNode insert\_into\_index(BpNode BpTree, unsigned char type, void\* K, int lines)

把K和指向文件的指针插入到B+树中

传入B+树（BpTree），K的地址

返回这棵B+树

BpNode insert\_in\_leaf(BpNode L, unsigned char type, void\* K, int lines)

将值插入B+树的叶子节点，按插入数据的类型来判断应该将数据插到哪个叶子节点

传入一棵B+树，数据K的类型，值以及行号

传出这棵B+树

BpNode insert\_in\_parent(BpNode root, unsigned char type, BpNode N, void\* K, BpNode NN, BpNode parent[])

插入一个root节点，首先判断节点N是否为root节点，如果是，将N的isroot设为0并新建一个节点作为root，如果不是，则新建一个节点作为root。

返回root