```
#ifndef AbstractNode04_h
#define AbstractNode04_h 1
/#define ORDER 2
#define OVERFULLKEYS (2*ORDER+2)
#define FULLKEYS (2*ORDER+1)
#define OVERFULLPOINTERS (2*ORDER+3)
#define FULLPOINTERS (2*ORDER+2)
//---------------------------------------------------------------------------
//Class AbstractNode defines the virtual interface of a B+AbstractNode
//---------------------------------------------------------------------------
class AbstractNode {
public:
    virtual void Print(ostream& o,int depth)=0;
    virtual AbstractNode* FindChild(int k)=0;
    virtual AbstractNode* Child(int k)=0;
    virtual void Insert(AbstractNode* K)=0;
    virtual bool IsOverFull(void)=0;
    virtual int FetchKey(int i)=0;
    virtual int FetchKeyCount(void)=0;
    virtual AbstractNode* SplitNHoist(void)=0;
    virtual bool IsHoisted(void)=0;
    virtual void ResetIsHoisted(void)=0;
};
#endif
```

Figure 1. class *AbstractNode*.

```
#ifndef Leaf04_h
#define Leaf04_h 1
class Leaf: public AbstractNode {
protected:
     int keycount;
     int key[OVERFULLKEYS];
public:
    Leaf();
    Leaf(int k);
    void Print(ostream& o,int depth);
    AbstractNode* FindChild(int k);
    AbstractNode* Child(int k);
    void Insert(AbstractNode* K);
    bool IsMember(int k);
    bool IsOverFull(void);
    int Index(int k);
    int FetchKey(int i);
    int FetchKeyCount(void);
    AbstractNode* SplitNHoist(void);
    bool IsHoisted(void);
    void ResetIsHoisted(void);
};
#endif
```

Figure 2. class *Leaf.*

```
#ifndef Node04_h
#define Node04_h 1
#include "Leaf04.h"
class Node : public Leaf {
      AbstractNode* child[OVERFULLPOINTERS];
       bool ishoisted;
public:
      Node();
      Node(AbstractNode* p1,int k1,AbstractNode* p2);
      AbstractNode* FindChild(int k);
      AbstractNode* Child(int k);
      void  Print(ostream& o,int depth);
      void  Insert(AbstractNode* K);
       bool IsOverFull(void);
      AbstractNode* SplitNHoist(void);
       bool IsHoisted(void);
      void  ResetIsHoisted(void);
};
#endif
```

Figure 3. class *Node*.

```
#ifndef Tree04_h
#define Tree04_h 1
#include "Node04.h"
class Tree {
      AbstractNode* root;
      AbstractNode* Insert(AbstractNode* N,int k);
      void  Print(ostream& o,AbstractNode* N,int depth);
public:
      Tree();
      ~Tree();
      void  Insert(int k);
      void  Print(ostream& o);
};
#endif
```

Figure 4. class *Tree*.

```
AbstractNode* Node::Insert(AbstractNode* n,int k)
1. if (!n) return new Leaf(k);
2. if (k ∈ n) return n;
3. return Merge(Insert(n->FindChild(k),k);
```

Figure 5. Function *Insert*.