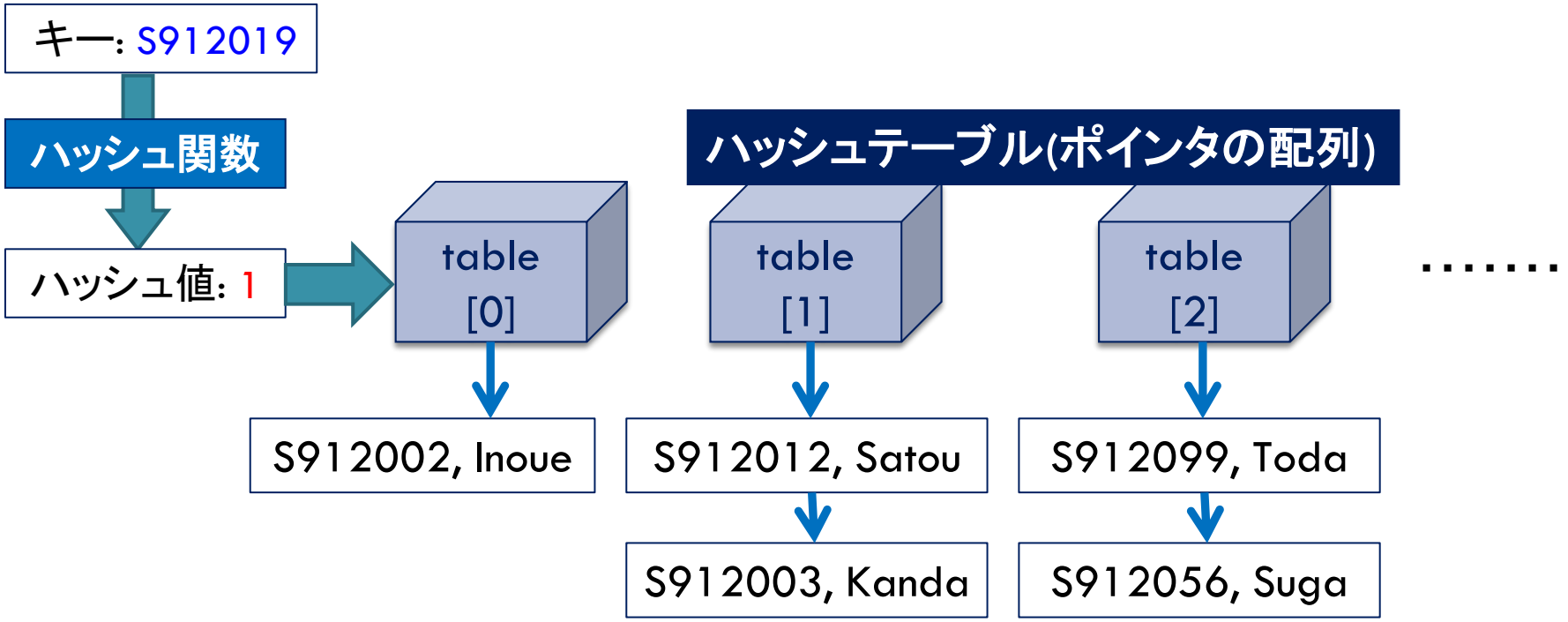
問題

第1引数にkey、第2引数にvalueをとり、
すでにkeyとvalueの組み合わせが登録されていたら、0を返し、
keyと別のvalueの組み合わせが登録されていたら、エラーを返し、
keyが未登録であれば、ハッシュ値hashvalueにおけるリストの先頭ノードtable[hashvalue]に
keyとvalueを格納した新しいノードを追加する関数

int Hash::insert(string key, string value)

を作成してください。HASHSIZEは7としてください。

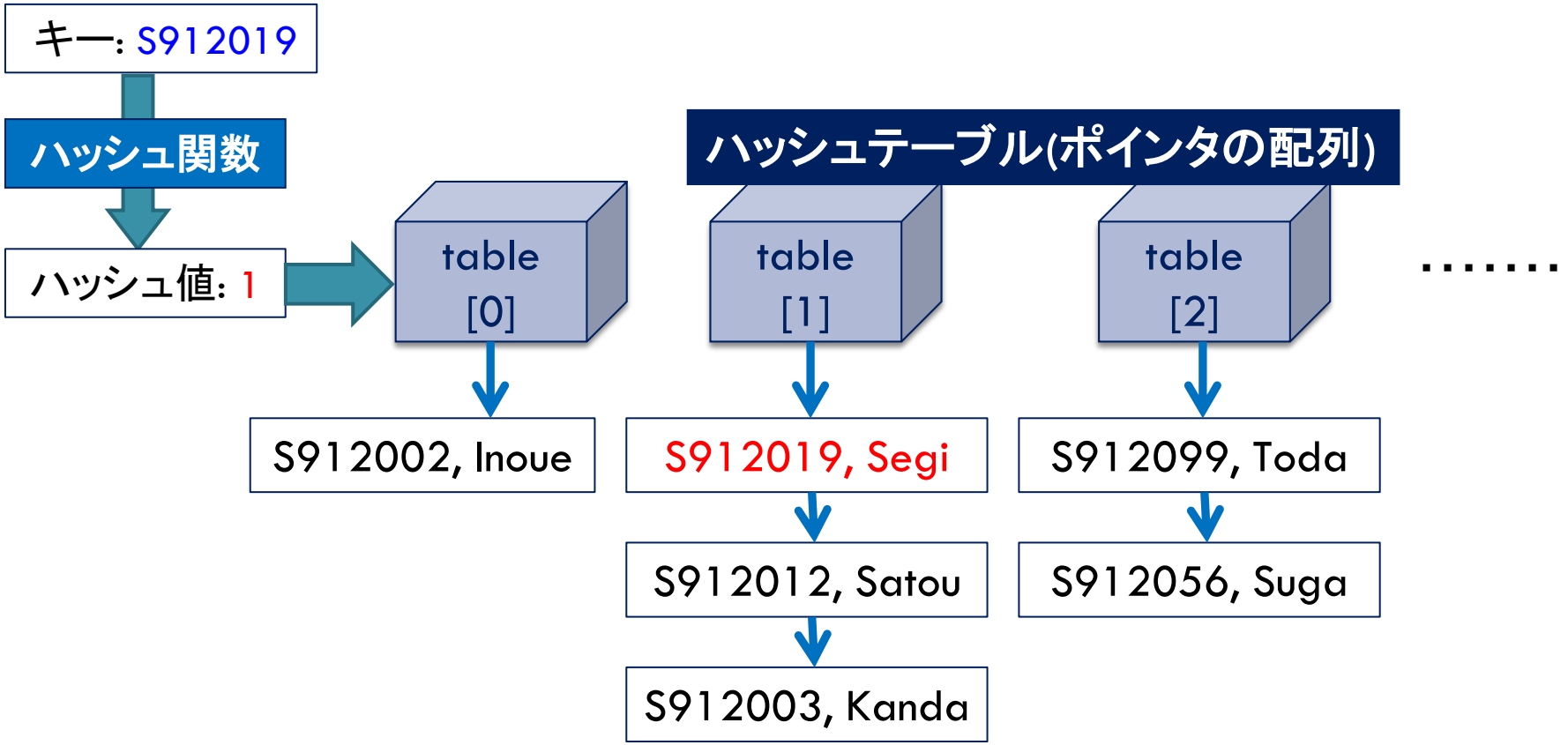


問題(続き)

第1引数にkey、第2引数にvalueをとり、
すでにkeyとvalueの組み合わせが登録されていたら、0を返し、
keyと別のvalueの組み合わせが登録されていたら、エラーを返し、
keyが未登録であれば、ハッシュ値hashvalueにおけるリストの先頭ノードtable[hashvalue]に
keyとvalueを格納した新しいノードを追加する関数

int Hash::insert(string key, string value)

を作成してください。HASHSIZEは7としてください。



問題(続き)

```
//mainは以下のプログラムを使用してください。
int main(void){
   Hash ht(HASHSIZE);                                    //ハッシュの作成
   string filename="data.csv";
   string line;
   ifstream fin(filename);
   if(!fin){                                             //エラー処理
      cerr << "エラー:ファイルを開けません" << filename << "¥n";
      exit(EXIT_FAILURE);
   }
   while(fin >> line){
      int index = line.find(",");                        //カンマの添え字がindexに返る
      string key = line.substr(0,index);                 //lineの0番目からindex-1番目までの文字列をkeyにコピー
      string value = line.substr(index+1);               //lineのindex+1番目から終わりまでの文字列をvalueにコピー
      ht.insert(key,value);                              //ハッシュにkeyとvalueのペアを追加
   }
   fin.close();
   cout << "¥n";
   ht.print();                                           //ハッシュの中身を出力

   string inputkey, inputvalue;
   cout << "¥ninput key ? ";
   cin >> inputkey;
   cout << "input value ? ";
   cin >> inputvalue;
   if(ht.insert(inputkey, inputvalue)==1){               //inputkeyとinputvalueを追加。
      cout << "WARNING:input keyとinput valueはすでに登録されています¥n";
   }else{
      ht.print();
   }
   return 0;
}
```

実行例1

```
comsv001% ./a.out

print Hash Table
0-> (S912002,Inoue)
1-> (S912012,Satou) -> (S912003,Kanda)
2-> (S912099,Toda) -> (S912056,Suga)
3-> (S912075,Taguchi)
4
5
6-> (S912001,Abe)

input key ? S912019
input value ? Segi
Print Hash Table
0-> (S912002,Inoue)
1-> (S912019,Segi) -> (S912012,Satou) -> (S912003,Kanda)
2-> (S912099,Toda) -> (S912056,Suga)
3-> (S912075,Taguchi)
4
5
6-> (S912001,Abe)
comsv001% ./a.out

print Hash Table
0-> (S912002,Inoue)
1-> (S912012,Satou) -> (S912003,Kanda)
2-> (S912099,Toda) -> (S912056,Suga)
3-> (S912075,Taguchi)
4
5
6-> (S912001,Abe)

input key ? S912002
input value ? Inoue
WARNING:input keyとinput valueはすでに登録されています
```

実行例2

```
comsv001% ./a.out

Print Hash Table
0-> (S912002,Inoue)
1-> (S912012,Satou) -> (S912003,Kanda)
2-> (S912099,Toda) -> (S912056,Suga)
3-> (S912075,Taguchi)
4
5
6-> (S912001,Abe)

input key ? S912002
input value ? Inohara
ERROR:key and different value have already been registered:S912002 Inohara Inoue
comsv001%
```

解答

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <string>
using namespace std;
#define HASHSIZE 7
class Hash{
private:
    class Node {
    public:
        string key;                                 // キー
        string value;                               // 値（バリュー）
        Node *next;                                 // 後続ノードへのポインタ(リスト構造)
        Node(){}
        Node(string k, string v, Node *n=NULL) {
            key=k; value=v; next=n;
        }
    };
    int size;                                       // ハッシュテーブルの大きさ
    Node **table;                                   // ハッシュ値が同じノードの先頭アドレス
    int get_hashvalue(string);                      // ハッシュ値の取得(ハッシュ関数)
public:
    Hash(int tablesize);
    Node* find(string key);
    string find_value(Node *findnode);
    string find_key(Node *findnode);
    int insert(string key, string value);
    void print();
};

// コンストラクタ（ハッシュテーブルのメモリ確保と初期化）
Hash::Hash(int tablesize){
    table=(Node **) new Node[tablesize];    //引数で指定された大きさのハッシュテーブルのメモリを確保
    size=tablesize;
    for(int i=0; i<tablesize; i++){
        table[i]=NULL;                              //NULLで初期化しておく
    }
}
```

```
// ハッシュ値の取得(ハッシュ関数)
int Hash::get_hashvalue(string key){
    unsigned int v=0;
    for(unsigned int i=0; i<key.length(); i++){
        v += key[i];                          //1文字ごとの文字コードを加算
    }
    return v%size;                            //ハッシュテーブルの大きさで割った余りを返す
}

// keyを保持するノードのポインタを戻す
Hash::Node* Hash::find(string key){
    int hashvalue=get_hashvalue(key);         //ハッシュ値の取得
    Node *p=table[hashvalue];                 //ハッシュテーブルに格納されたリストの先頭ノードのポインタを取得
    if(p==NULL){                              //NULLの場合
        return NULL;                          //NULLを返す(keyは登録されていない)
    }
    while(p !=NULL){                          //pがリストの末端まで行ききっていない場合
        if(p->key==key){                      //pが指しているノードのkeyが引数と一致していれば
            return p;                         //pを返す(keyを保持するノードを指すポインタがp)
        }
        p=p->next;                            //次のノードを調べる
    }
    return NULL;                              //リストにkeyが存在しないため、keyは登録されていない
}

// key値に対応したnodepointerを用いてvalueを戻す
string Hash::find_value(Node *nodepointer){
    return nodepointer->value;
}

// nodepointerのkeyを戻す
string Hash::find_key(Node *nodepointer){
    return nodepointer->key;
}
```

```
// keyとvalueの組合せを追加
int Hash::insert(string key, string value){
    // 同じキー値を持つデータは二重登録しない
    Node *findnode=find(key);
    if(findnode !=NULL){
        if(find_value(findnode)==value){
            return 1;
        }else{
            cerr << "ERROR:key and different value have already been registered:" << key << " " << value << " " <<
find_value(findnode) << endl;
            exit(EXIT_FAILURE);
       }
    }
    int hashvalue=get_hashvalue(key);                  // 追加するkeyのハッシュ値を取得
    Node *head=table[hashvalue];                       // 当該ハッシュ値のtableに格納されているリストの先頭を取得
    table[hashvalue]=new Node(key, value, head);  // リストの先頭にkeyとvalueを追加。旧リストの先頭をnextが指す
    return 0;
}

void Hash::print(){
    cout << "Print Hash Table¥n";
    for(int i=0; i<size; i++){                         //table[0]からtable[size-1]が出力対象
        Node *p=table[i];                              //リストの大元のノードのポインタを取得
        cout << i;
        while (p != NULL) {                            //pがNULLでない場合
            cout << "-> (" << p->key << "," << p->value << ") ";
            p = p->next;                               //次のノードを調べる
        }
        cout << "¥n";
    }
}
```

```
int main(void){
   Hash ht(HASHSIZE);                          //ハッシュの作成
   string filename="data.csv";
   string line;
   ifstream fin(filename);
   if(!fin){                                   //エラー処理
      cerr << "エラー:ファイルを開けません" << filename << "¥n";
      exit(EXIT_FAILURE);
   }
   while(fin >> line){
      int index = line.find(",");              //カンマの添え字がindexに返る
      string key = line.substr(0,index);       //lineの0番目からindex-1番目までの文字列をkeyにコピー
      string value = line.substr(index+1);     //lineのindex+1番目から終わりまでの文字列をvalueにコピー
      ht.insert(key,value);                    //ハッシュにkeyとvalueのペアを追加
   }
   fin.close();
   cout << "¥n";
   ht.print();                                 //ハッシュの中身を出力

   string inputkey, inputvalue;
   cout << "¥ninput key ? ";
   cin >> inputkey;
   cout << "input value ? ";
   cin >> inputvalue;
   if(ht.insert(inputkey, inputvalue)==1){     //inputkeyとinputvalueを追加。
      cout << "WARNING:input keyとinput valueはすでに登録されています¥n";
   }else{
      ht.print();
   }
   return 0;
}
```