5．1

（a）

///////////////////////////////////////////////////////////////

///////////////// solution one /////////////////

//////////////////////////////////////////////////////////////

typedef struct bucket \* \_bucket;

\_bucket \* rehash(\_bucket \* );

int bucketnum=0;

int size=SIZE;

void insert(string key,void \* binding,struct bucket \* table)

{

int index;

bucketnum++;

if(bucketnum/size>2)

{

size\*=2;

table=rehash(table);

}

index=hash(key)%size;

table[index]=Bucket(key,binding,table[index]);

}

\_bucket \* rehash(\_bucket \* table)

{

\_bucket \* newTable=(\_bucket \*)malloc(sizeof(\_bucket \*)\*size);

int i;

for(i=0;i<bucketnum;i++)

{

insert(table[i]->key,table[i]->binding,newTable);

}

free(table);

return newTable;

}

////////////////////////////////////////////////////////////////////////

////////////////////// solution two ///////////////////////

/////////////////////////////////////////////////////////////////////////

int size=SIZE;

bool need\_enlarge()

{

int count=0;

for(int i=0;i<size;i++)

{

struct bucket \* b=checked\_malloc(sizeof(\*b));

for(b=(\*table)[i];b;b=b->next)

count++;

}

if(count/SIZE>2)

return true;

else

return false;

}

void enlarge()

{

size\*=2;

struct bucket \*\*temp=table;

table=malloc(sizeof(struct bucket \*)\*size);

for(int i=0;i<size/2;i++)

{

struct bucket \*b=checked\_malloc(sizeof(\*b));

for(b=(\*temp)[i];b;b=b->next)

{

insert(b->key,b->binding);

}

}

free(temp);

}

void insert(string key,void \* binding)

{

int index=hash(key)%size;

(\*table)[index]=bucket(key,binding,(\*table)[index]);

}

/////////////////////////////////////////////////////////////////////////

///////////// solution three ///////////////////////////////

/////////////////////////////////////////////////////////////////////////

int bucketNum=0;

struct bucket \* table[SIZE];

struct bucket \*\*tableOld;

int size=SIZE;

void insert(string key,void \* binding)

{

bucketNum++;

if(bucketNum/size>2)

{

size\*=2;

bucketNum=0;

tableOld=malloc(sizeof(\*bucket)\*size/2);

for(int i=0;i<size/2;i++)

tableOld[i]=table[i];

table=malloc(sizeof(struct bucket \*)\*size);

for(int i=0;i<size/2;i++)

{

for(struct bucket \*p=tableOld[i];p!=NULL;p=p->next)

{

insert(p->key,p->binding);

}

free(p);

}

}

else

{

int index=hash(key)%size;

table[index]=bucket(key,binding,table[index]);

}

}

（b）

void insert(string key,void \* binding,int num)

{

int index=hash(key)%SIZE;

table[num][index]=Bucket(key,binding,table[index]);

}

void \* lookup(string key,int num)

{

int index=hash(key)%SIZE;

struct bucket \* b;

for(b=table[num][index];b;b=b->next)

if(0==strcmp(b->key,key))return b->binding;

return NULL;

}