```c
/* ClosedHashTest.c */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <malloc.h>
#include "ClosedHash.h"

typedef struct
{
    int number;
    char **name;
}Student;

static int IntHash(const void *e)
{
    int key = *(int *)e;
    return (key % 31);
}
static int IntCollide(int hashKey, int count)
{
    return ((hashKey + count * count) % 31);
}

static int IntCmp(const void *keyAddr, const void *dataAddr)
{
    int *p1 = (int *)keyAddr;
    int *p2 = (int *)dataAddr;
    return (*p1 - *p2);
}
static void StudentFree(void *keyAddr)
{
    int *key = (int *)keyAddr;
    if (*key != -1)
    {
        Student *stu = (Student *)((char *)keyAddr + sizeof(int));
        char *name = *(char **)(stu->name);
        free(name);
    }
}

int main()
{
    HASH intHash;
    HashNew(&intHash, 31, sizeof(int), sizeof(Student), IntHash, IntCollide, IntCmp,
    StudentFree);
    printf("intHash capacity is %d\n", HashCapacity(&intHash));
    char *name1 = strdup("pc");
    int key1 = 1;
    char *name2 = strdup("jerry");
    int key2 = 32;
    char *name3 = strdup("hada");
    int key3 = 3;
    char *name4 = strdup("sunanzhi");
    int key4 = 4;
    char *name5 = strdup("zhangyouhe");
    int key5 = 5;
    char *name6 = strdup("xiejinying");
    int key6 = 6;
    char *name7 = strdup("yuzhiqiang");
    int key7 = 7;
    char *name8 = strdup("liyunlong");
    int key8 = 8;
    char *name9 = strdup("luyuebin");
    int key9 = 9;
    char *name10 = strdup("lihui");
    int key10 = 10;
    char *name11 = strdup("renwenjie");
    int key11 = 11;
    char *name12 = strdup("chenzhaojie");
    int key12 = 12;
    Student s1 = {key1, &name1};
    Student s2 = {key2, &name2};
    Student s3 = {key3, &name3};
    Student s4 = {key4, &name4};
    Student s5 = {key5, &name5};
    Student s6 = {key6, &name6};
    Student s7 = {key7, &name7};
    Student s8 = {key8, &name8};
    Student s9 = {key9, &name9};
    Student s10 = {key10, &name10};
    Student s11 = {key11, &name11};
    Student s12 = {key12, &name12};
    HashPut(&intHash, &key1, &s1);
    HashPut(&intHash, &key2, &s2);
    HashPut(&intHash, &key3, &s3);
    HashPut(&intHash, &key4, &s4);
    HashPut(&intHash, &key5, &s5);
    HashPut(&intHash, &key6, &s6);
    HashPut(&intHash, &key7, &s7);
    HashPut(&intHash, &key8, &s8);
    HashPut(&intHash, &key9, &s9);
    HashPut(&intHash, &key10, &s10);
    HashPut(&intHash, &key11, &s11);
    HashPut(&intHash, &key12, &s12);
    printf("intHash size is %d\n", HashSize(&intHash));
    void *sGet = HashGet(&intHash, &key1);
    if (NULL != sGet)
    {
        Student *stu = (Student *)((char *)sGet + sizeof(int));
        printf("get student %s from intHash\n", *(char **)stu->name);
    }
    else
    {
        printf("key %d is not in intHash\n", key1);
    }
    HashRemove(&intHash, &key1);
    sGet = HashGet(&intHash, &key1);
    if (NULL != sGet)
    {
        Student *stu = (Student *)((char *)sGet + sizeof(int));
        printf("get student %s from intHash\n", *(char **)stu->name);
    }
    else
    {
        printf("key %d is not in intHash\n", key1);
    }
    printf("intHash size is %d\n", HashSize(&intHash));
    HashDispose(&intHash);
    return 0;
}
```