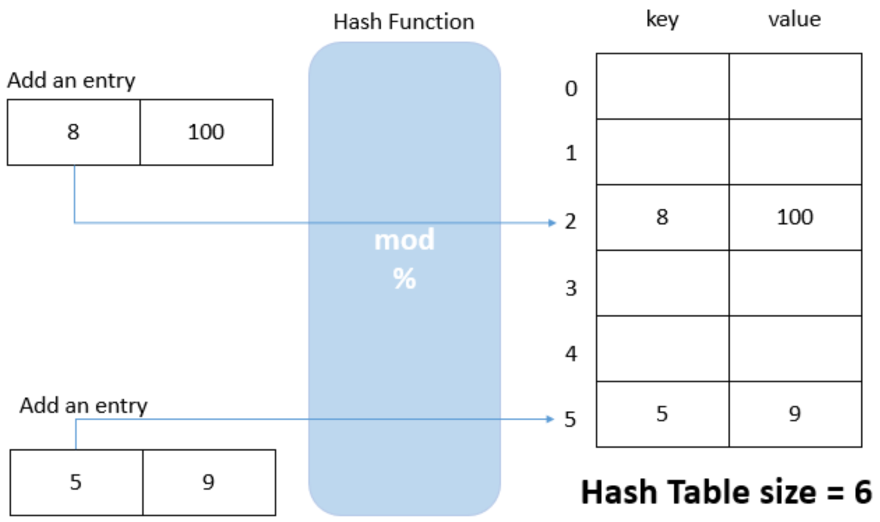
Introduction to Computers and Programming LAB-9 2016/11/30

- ✧ The output must be in our sample output format.
- ✧ You can raise your hand to demo once you finish a program.
- ✧ The bonus question is for this lab only. If you cannot finish question 1 to 3 in time, you are allowed to demo them at next lab hours.
- ✧ TAs will update lab records every Monday after the lab hours in the link: http://goo.gl/ZVJu2Y

1. Hash Table



A hash table is a data structure that can use a "key" to directly search the position where an entry is store. Given a function which maps a key to a position in the hash table. It speeds up the times a search an entry. The function is called hash function, and the table storing these entries in particular orders is the hash table.

In this question, given **the size of the hash table N (< = 10)**, implement a simple hash table. An entry contains two integers, which are **(key, value)**. The hash function in defined to be a integer division (%), which is $\mathrm{h(x)} = KEY \ \% \ N$. There are three operations you have to implement:

(1) ***Add*: Hash an input entry (KEY, VALUE) to the hash table**. The position of it is determined by the hash function $\mathrm{h(x)} = KEY \ \% \ N$. If the position is unavailable, add 1 to the current position until find an appropriate position (search for a pass). If the program goes through all the table and still can't find a position for the entry, then show "*[INFO] The space is not enough.*"; otherwise, show "*[INFO] The entry KEY has been added.*"

(2) ***Search and Remove:* Input the KEY of the entry, search the entry by the method same as *add* and remove it from the hash table if it is in the table**. Show "*[INFO] The entry KEY has been removed.*" if the deletion is successful; otherwise, show "[INFO] The KEY is not in the hash table."

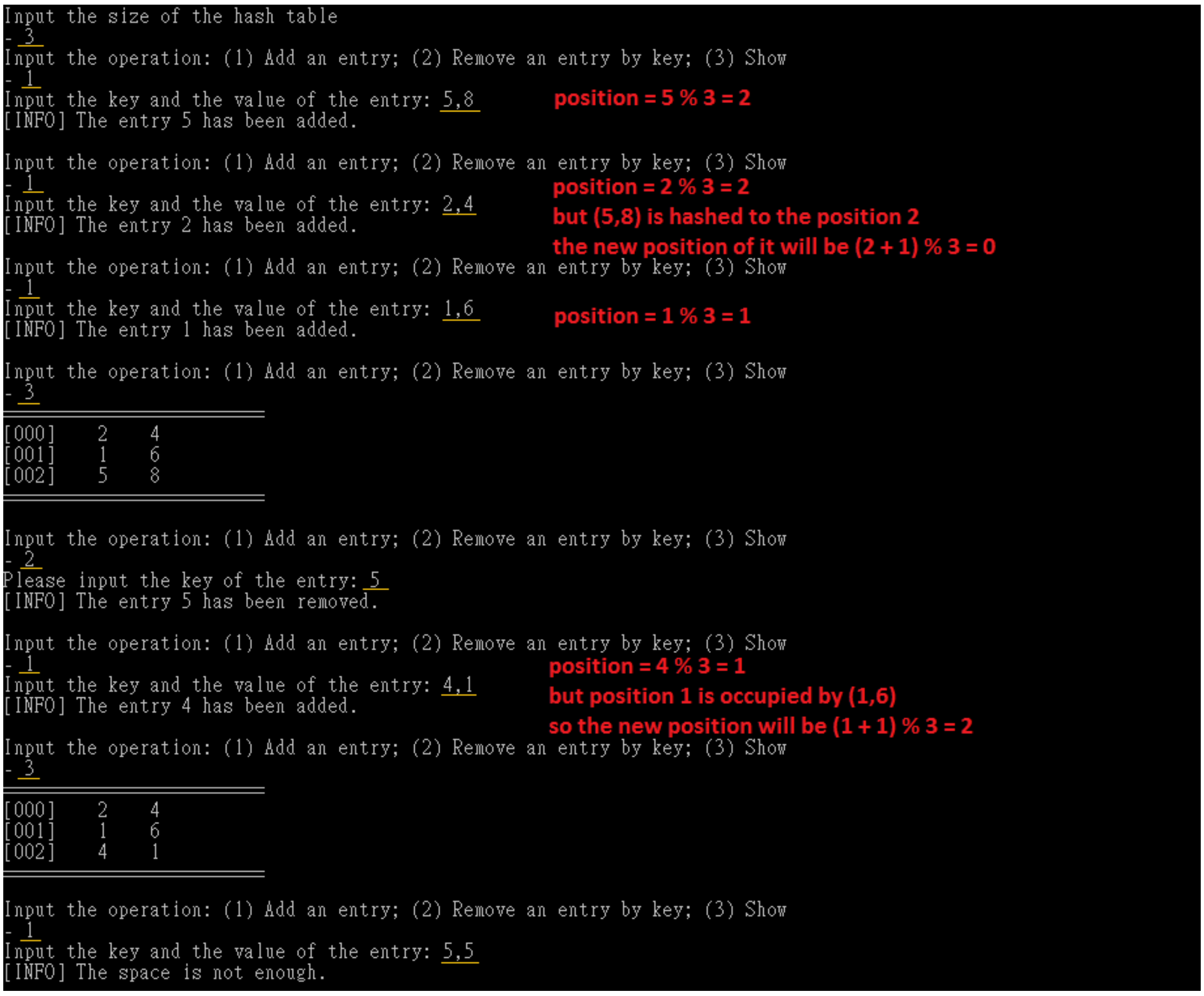
(3) ***Show*: Show the hash table from position 0 to N-1**. The format is "[00N]   KEY   VALUE"
If an entry is NULL in the table, KEY=-1, VALUE=-1

Please finish the following codes. The structure ***HashEntry*** represent the entry in the hash table. The structure ***HashTable*** represent the hash table contains N hash entries.

```c
#include<stdio.h>
#define MAX_TABLE_SZ 10
typedef struct
{
    int key;
    int value;
}HashEntry;

typedef struct
{
    int table_size;
    HashEntry HE[MAX_TABLE_SZ];
}HashTable;

int main(void)
{
    //write your codes here
    return 0;
}
```



```
Input the size of the hash table
- 3
Input the operation: (1) Add an entry; (2) Remove an entry by key; (3) Show
- 1
Input the key and the value of the entry: 5,8         position = 5 % 3 = 2
[INFO] The entry 5 has been added.

Input the operation: (1) Add an entry; (2) Remove an entry by key; (3) Show
- 1                                                   position = 2 % 3 = 2
Input the key and the value of the entry: 2,4         but (5,8) is hashed to the position 2
[INFO] The entry 2 has been added.
                                                      the new position of it will be (2 + 1) % 3 = 0
Input the operation: (1) Add an entry; (2) Remove an entry by key; (3) Show
- 1
Input the key and the value of the entry: 1,6         position = 1 % 3 = 1
[INFO] The entry 1 has been added.

Input the operation: (1) Add an entry; (2) Remove an entry by key; (3) Show
- 3
=========================
[000]    2    4
[001]    1    6
[002]    5    8
=========================

Input the operation: (1) Add an entry; (2) Remove an entry by key; (3) Show
- 2
Please input the key of the entry: 5
[INFO] The entry 5 has been removed.

Input the operation: (1) Add an entry; (2) Remove an entry by key; (3) Show
- 1                                                   position = 4 % 3 = 1
Input the key and the value of the entry: 4,1         but position 1 is occupied by (1,6)
[INFO] The entry 4 has been added.
                                                      so the new position will be (1 + 1) % 3 = 2
Input the operation: (1) Add an entry; (2) Remove an entry by key; (3) Show
- 3
=========================
[000]    2    4
[001]    1    6
[002]    4    1
=========================

Input the operation: (1) Add an entry; (2) Remove an entry by key; (3) Show
- 1
Input the key and the value of the entry: 5,5
[INFO] The space is not enough.
```

2. String Operator

This question is designed to let you practice some common used function in the library <string.h>. In this question, we have 6 char array (*str0~str5*) to save the *input* string. Like the stack operation we made last week, here are some function that you need to finish:

Format: function (command)

(1) **assign** (assign *str1 input*): give the *input* string into the empty *str1*

(2) **clear** (clear *str1*): clear the string in *str1*

(3) **copy** (copy *str1 str2/input*): copy the string in *str2* (or the *input* string) into *str1*.

(4) **cat(concatenate)** (cat *str1 str2/input*): concatenate the string in *str1* with the string in *str2* (or the *input* string) and save in *str1*.

(5) **show** (show): show all the str that is NOT empty

(6) **cmp(compare)** (cmp *str1 str2/input*): compare whether the string in *str1* and the string in *str2* (or the *input* string) are the same or not.

Some rule:

(1) The second position in the command is always the char array *str* in the function: **assign**, **clear**, **copy**, **cat**, and **cmp**.(The first position is the function: **assign**, **clear**, **copy**, **cat**, and **cmp**.) Therefore, you should print "invalid" when the second position in the command is not char array *str*.

(2) If the input char array *str*i is out of range (i greater than 5), then print "invalid". E.g. assign *str6 abc*. => *str6* is out of the range of char array str.

(3) When assign a string into a char array *str*, the char array *str* must be empty, or else you need to print "*str* has already been assign".

(4) Cannot concatenate or copy an empty string to a char array *str*, you should handle it and print "Cannot concatenate/copy from an empty string!"

(5) If compare with two empty char array *str0* and *str1*, we define they are the same in this question.

(6) We will NOT test the empty *input* string and the *input* string with space. Therefore, you don't need to handle this kind of problem.

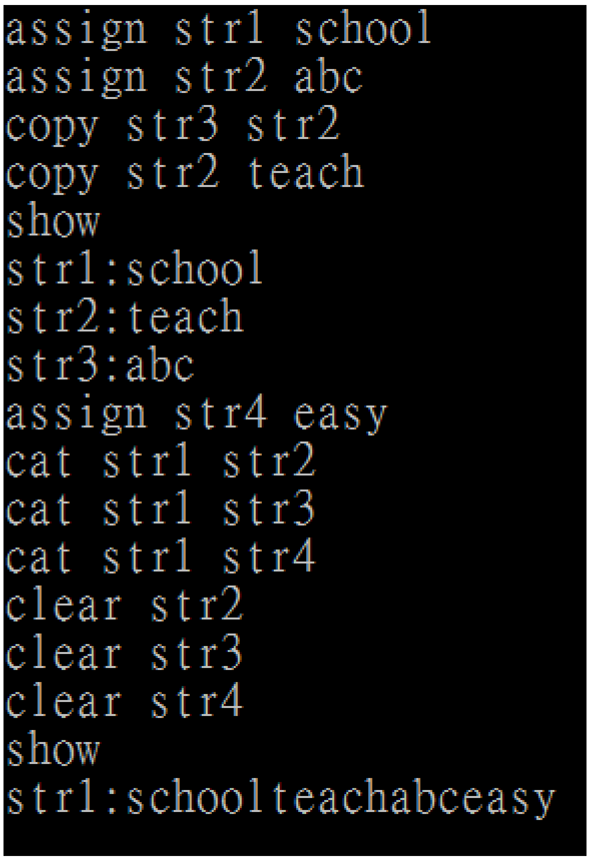
Examples:

(1) The correct output of the function assign, copy, cat, clear, and show.

(2) The correct output of the function compare.

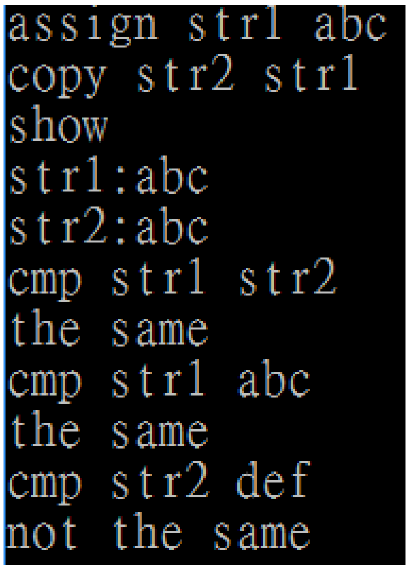
(3) Some exception that you should care about.

(1)



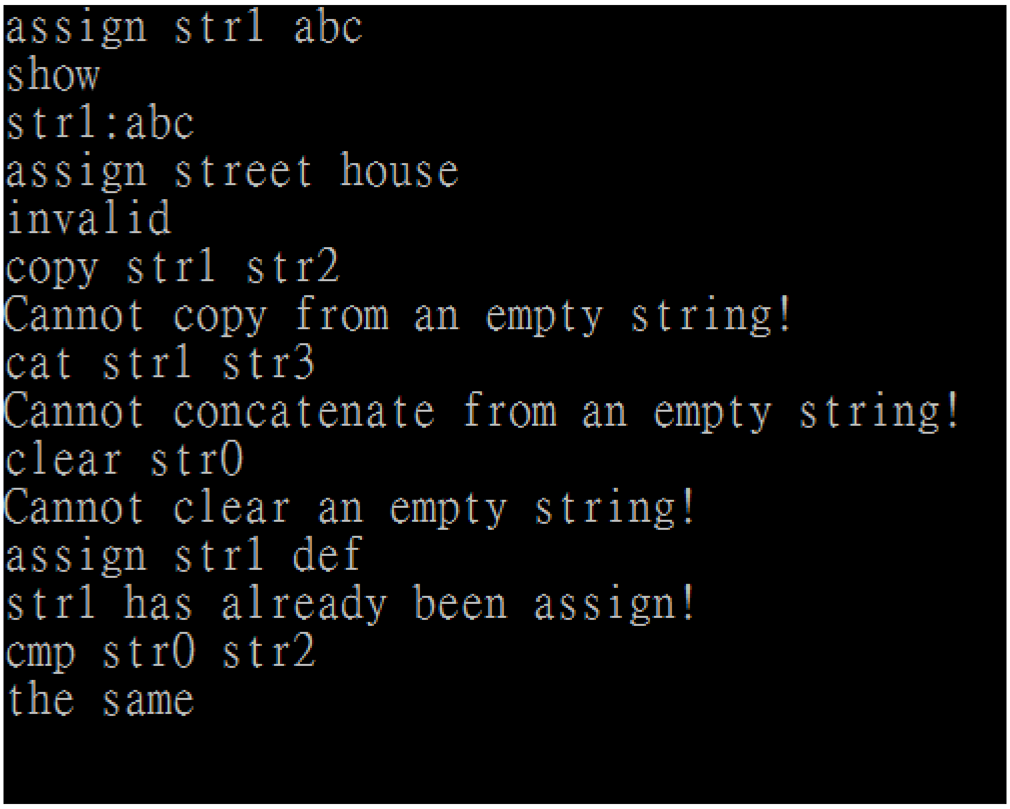
```
assign str1 school
assign str2 abc
copy str3 str2
copy str2 teach
show
str1:school
str2:teach
str3:abc
assign str4 easy
cat str1 str2
cat str1 str3
cat str1 str4
clear str2
clear str3
clear str4
show
str1:schoolteachabceasy
```

(2)



```
assign str1 abc
copy str2 str1
show
str1:abc
str2:abc
cmp str1 str2
the same
cmp str1 abc
the same
cmp str2 def
not the same
```

(3)



```
assign str1 abc
show
str1:abc
assign street house
invalid
copy str1 str2
Cannot copy from an empty string!
cat str1 str3
Cannot concatenate from an empty string!
clear str0
Cannot clear an empty string!
assign str1 def
str1 has already been assign!
cmp str0 str2
the same
```

3. Simple Command Parser

Please implement a simple command parser which can parse the following command.

There are 3 types of command:

type 0: with no argument

exit: When user input this command, exit the program.

type 1: with one argument

view tableName

delete tableName

Action: Show its argument when user input this type command. (Please refer to example.)

type 2: with one preposition, one argument list and one tail argument

create a,b,c… **as** tableName

insert 1,2,3… **into** tableName

remove 1,2,3… **from** tableName

Action: Show its arguments when user input this type command. (Please refer to example.)

Yellow is user input, and red is output.

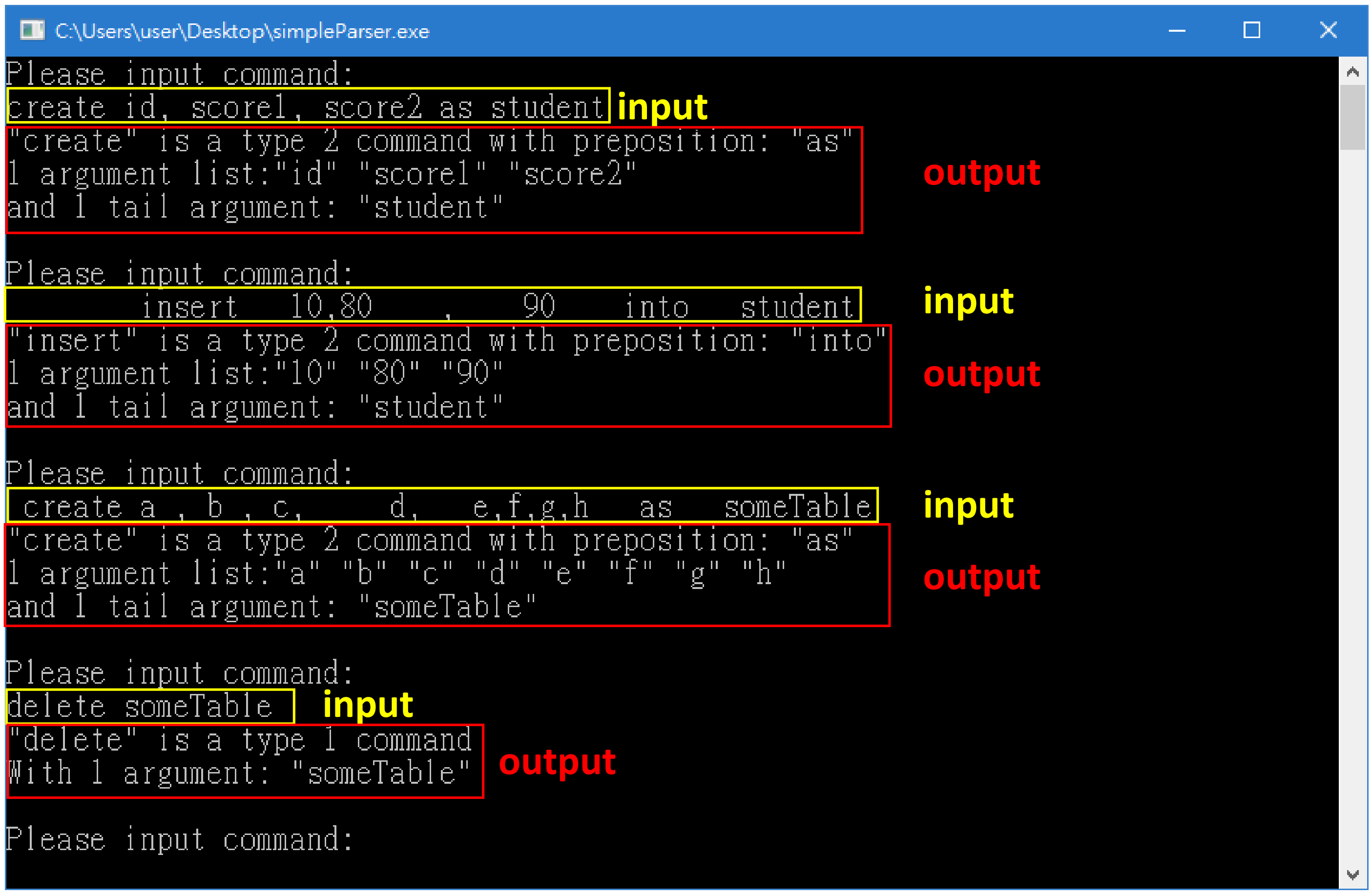
```
Please input command:
create id, score1, score2 as student
"create" is a type 2 command with preposition: "as"
1 argument list:"id" "score1" "score2"
and 1 tail argument: "student"

Please input command:
  insert      103    ,100 ,92    into      student
"insert" is a type 2 command with preposition: "into"
1 argument list:"103" "100" "92"
and 1 tail argument: "student"

Please input command:
                                   remove                       83    from    tmpTable
"remove" is a type 2 command with preposition: "from"
1 argument list:"83"
and 1 tail argument: "tmpTable"

Please input command:
delete                                                   tmpTable
"delete" is a type 1 command
With 1 argument: "tmpTable"

Please input command:
                                                                                     view       aaa
"view" is a type 1 command
With 1 argument: "aaa"
```



```
Please input command:
create id, score1, score2 as student
"create" is a type 2 command with preposition: "as"
1 argument list:"id" "score1" "score2"
and 1 tail argument: "student"

Please input command:
        insert   10,80      ,     90     into    student
"insert" is a type 2 command with preposition: "into"
1 argument list:"10" "80" "90"
and 1 tail argument: "student"

Please input command:
 create a , b , c,     d,    e,f,g,h   as    someTable
"create" is a type 2 command with preposition: "as"
1 argument list:"a" "b" "c" "d" "e" "f" "g" "h"
and 1 tail argument: "someTable"

Please input command:
delete someTable
"delete" is a type 1 command
With 1 argument: "someTable"

Please input command:
```

Note:

1. Your program need to repeat until user input exit.
2. The space between command and argument **is valid**, and user can also add space before command.
3. You have to remove all the **space** in argument and them print them.
4. When your program gets an unknown command such as “xxx 1234”,
   it should print “Unknown command "xxx".” And you don’t need to handle other invalid input.
5. **The libraries allowed to use are <stdio.h>, <stdlib.h>, <string.h>**

4. (BONUS) Simple Database

Please use the parser you implemented in question 3 to make a simple database.

There are 3 types of command:

type 0: with no argument

exit: When user input this command, exit the program.

type 1: with one argument

view tableX: show the content of tableX

(Show an error message when table is not found)

delete tableX: delete tableX

(Show an error message when table is not found)

type 2: with one preposition, one argument list and one tail argument

> **create** a,b,c **as** tableX: create a table with 3 columns a, b, c
>
> **(Show an error message when table is already existed or there are already 10 tables)**
>
> **insert** 1,2,3 **into** tableX: insert 1, 2, 3 into tableX
>
> **(Show an error message when input not match column count, or table is not found)**
>
> **remove** 1,2,3 **from** tableX: remove 1, 2, 3 from tableX
>
> **(Show how many rows it removed and show an error message when table is not found)**

Note:

1. After each operation, show how many tables are there in this database.
2. The maximum table count is 10, and each table can only have at most 100 rows and 10 columns.
3. All the data in this database is **double**.
4. **The libraries allowed to use are <stdio.h>, <stdlib.h>, <string.h>**

Example:

Example1:

```
There are 0 table(s) in the database.
create a, b , c as table1
Success !!

There are 1 table(s) in the database.
create a, b , c,d    as table1
Error: Table "table1" is already exist!!

There are 1 table(s) in the database.
create    a , b ,c, d as table2
Success !!

There are 2 table(s) in the database.
insert    1, 2 , 3 into table2
Error: Column count not match!!

There are 2 table(s) in the database.
insert    1, 2 , 3 into table1
Success !!

There are 2 table(s) in the database.
insert    4, 5 , 6 into table1
Success !!

There are 2 table(s) in the database.
insert    4, 5 , 6 into table1
Success !!

There are 2 table(s) in the database.
view table2
```

Example2:

```
There are 0 table(s) in the database.
create a,b,c as table1
Success !!

There are 1 table(s) in the database.
insert 1,1,1 into table1
Success !!

There are 1 table(s) in the database.
insert 1.01,1.01,1.01 into table1
Success !!

There are 1 table(s) in the database.
insert 1.5,1.5,1.5 into table1
Success !!

There are 1 table(s) in the database.
insert 1.5,1.5,1.5 into table1
Success !!

There are 1 table(s) in the database.
view table1
  a        b        c
 1.00     1.00     1.00
 1.01     1.01     1.01
 1.50     1.50     1.50
 1.50     1.50     1.50

There are 1 table(s) in the database.
delete table1
```

```
a          b        c          d

There are 2 table(s) in the database.
view table1
   a          b         c
 1.00      2.00      3.00
 4.00      5.00      6.00
 4.00      5.00      6.00

There are 2 table(s) in the database.
remove 4,5,6 from table1
Success remove 2 row(s) from table1!!

There are 2 table(s) in the database.
view table1
   a          b         c
 1.00      2.00      3.00

There are 2 table(s) in the database.
exit
```

```
Success !!

There are 0 table(s) in the database.
show table1
Unknown command "show"

There are 0 table(s) in the database.
view table1
Error: Can not find table: "table1"

There are 0 table(s) in the database.
create a as table1
Success !!

There are 1 table(s) in the database.
view table1
a

There are 1 table(s) in the database.
exit
```