HOMEWORK 8

ELSY FERNANDES

STU ID: 1001602253

↳ written part

↳ Empty table

Drawing

Index		
0	NULL	→ 0x...
1	NULL	→ 0x12...
2	NULL	→ 0x123...
3	NULL	→ 0x1234
4	NULL	→ 0x12345
5	NULL	→ 0x12345
6	NULL	→ 0x13456
7	NULL	→ 0x1
8	NULL	→ 0x
9	NULL	→ 0x
6	~~NULL~~	

Code → creates an Empty table

```
// Initialize a Structure
Struct Item
{
 Char EnglishWord[100];
 Char SpanishWord[1000];
 };
int Size = 0;
// allocate a Memory for a Empty table
Sturct Item *dictionary
      = (Struct Item*) malloc
          ( Size * Size of (Struct Item));
for (i=0 ; i<Size ; i++) {
Strncpy (dictionary[i].englishwold ,
          "    ", 100);
Strncpy (dictionary[i].Spanishwold ,
          "    ", 1000);
 }.
```

2. Table with Entry ["horse", "caballo"] at index 1 and Cat ["cat, "el gato"] at index 3.

Note: Drawing and code is aper my code in dictionary.C

Drawing

Before, calculate haskey for horse and Cat to Insert in table.

1	horse	Caballo
2	NULL	NULL
3	Cat	elgato
4	NULL	NULL
5	NULL	NULL
6	NULL	NULL
7	NULL	NULL
8	NULL	NULL
9	NULL	NULL
10	NULL	NULL

code

```
int Size=10;

Struct Item Insert (Struct Item dictionary[ ],
    Struct Item data, int size) {
int stest = Strcmp (destination.englishwold, " ");
int test2 = Strcmp (destination.englishwold, "DELETED"
if (test==0) && (test1!=0)
dictionary[key].englishwold = data.Englishwold;
dictionary[key].Spanishwold = data.Spanishwold;
}
```

↓

This Function checks if the hash table at a particular index is Empty and if hash table at a particular index was deleted previously.

→ if hash table is Empty and if it was not deleted previously then it will insert a [englishwold at spanishwold] at a index.

3) Table after user operation i Cat gato

Drawing

1	horse	Caballo
2	NULL	—
3	Cat	elgato; gato
4	—	—
5	—	—
6	—	—
7	—	—
8	—	—
9	—	—
10	—	—

It does need to reallocate memory

code.

Now insert i Cat gato

~~data.dictionary[key] = cat~~

~~data~~. data.Enghshwold = Cat

data.Spamshwold = gato

// do a insert operation

```
Struct Item* insert (Struct Item dictionary[],
    Struct item data, init size)
{  int test1 = strcmp(destination.enghshwd,"");
   int test2 = strcmp(destination.enghshwold,
                      "DELETED");
   if (test==0) && (test1!=0)
dictionary[key].enghshwold = data.englishwold;
dictionary[key].Spanishwold = data.Spamshwold
//When the word is already present
int test = strcmp (destmation.enghshwold, source.
                   enghshwold));
int test1 = Strcmp(destination.enghshwold, "DELETED");
 if   (test==0) &(test1=0);
//append a spanish word using strncat
Strncat (~~destmation~~ dictionary. Spanishwold, ";", 1);
Strncat (dictionary.Spanishwold, ~~source~~ data.
         Spamshwold, Strlen(data.Spanishwold));
```

It appends the string from data.spanishwold to dictionary.Spanish word using String concatenation.

4) Table after deleting the Item at Index 1.

Drawing

1	DELETED	DELETED
2	—	—
3	Cat	elgato; gato
4	—	—
5	—	—
6	—	—
7	—	—
8	—	—
9	—	—
10	—	—

Code.

Given wordcha English[]; delete Cat

```
int delete (char english[], Struct item
        dictionary[], int Size)
{
 in text = Strcmp(dictionary.englishwold, englis
int text2 = Strcmp(destination.englishwold,
                    "DELETED")
if (text==0) && (text1!=0)
// If given wold is already in the dictionary
   then delete the wold.
Strncpy(dictionary[key].englishwold, "DELETED",
                    100)
Strncpy(dictionary[key].spanishwold, "DELETED"
                    1000)
}
```

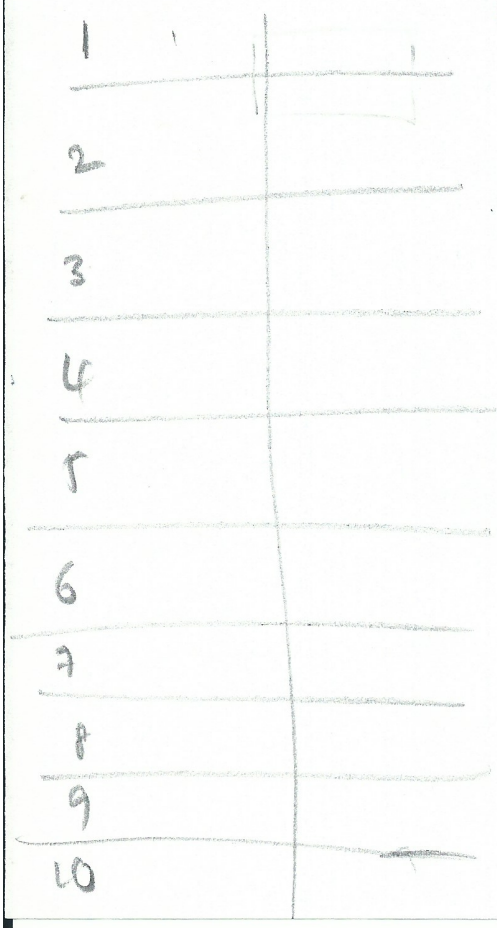
Note: Table value at index 1 is deleted and marked with "DELETED"

5) freed table. Here include code that frees the memory of needed.

↳ I had done dynamic memory allocation for the dictionary table. (Drawing 1).

Drawing

Memory may be available for further allocation. it will still look like how it is, until there is new data.



freed the allocation

```
int main()
{
free(dictionary);
int main()
{
Struct item
{
char Englishword[100];
Char Spanishwodd[1000];
};
int main() {
int size=0;
// allocate a memory for a
   Empty table
Struct Item *dictionary
= (Struct Item*) malloc
   (size of (Struct Item));
- - - -
- - -
free(dictionary);
```