Simple Hash Table

Homework Project

Description

雜湊表（Hash table，也叫哈希表），是根據鍵（Key）而直接查詢在記憶體儲存位置的資料結構。通過計算出一個鍵值的函數，將所需查詢的數據映射到表中一個位置來讓人查詢，這加快了查找速度。這個映射函數稱做雜湊函數，存放記錄的數組稱做雜湊表。

Procedure

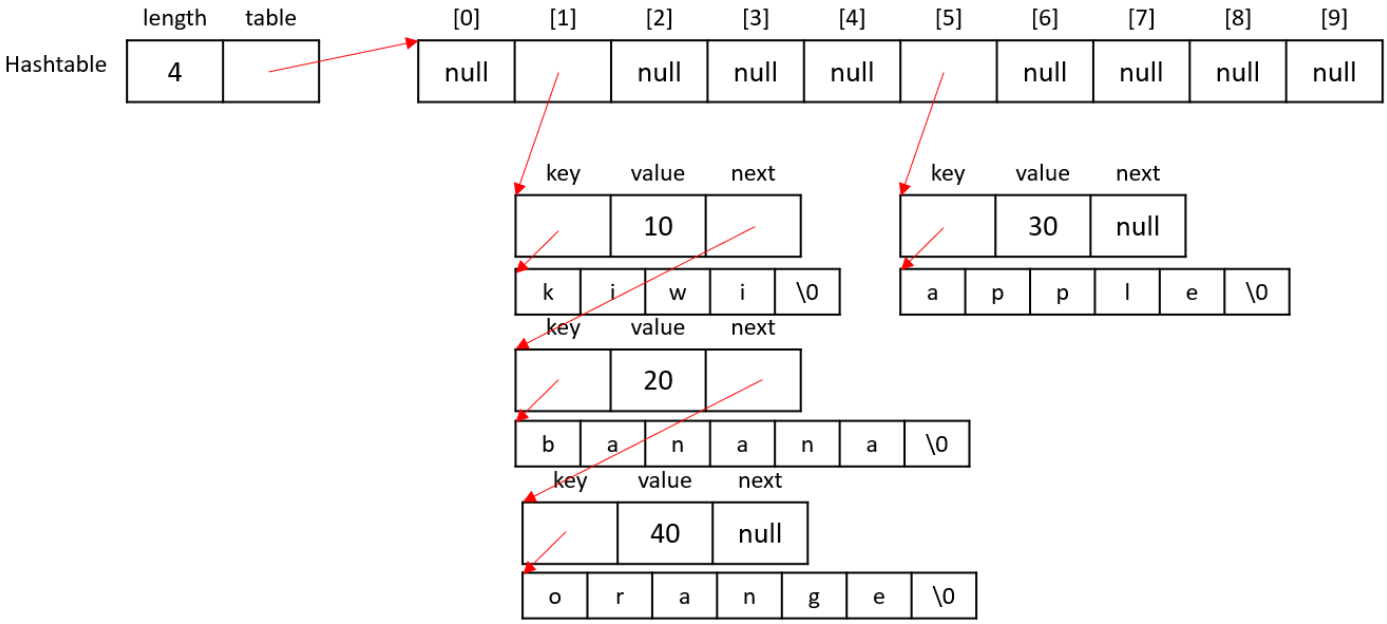
本作業同學需實作一個字串對應整數的雜湊表，建立在 Hashtable 結構中，其定義如下:

```
typedef struct hashtable
{
    Element **table; // the hash function is MurmurOAAT32 mod TABLE_SIZE
    int length; // count of elements
} Hashtable;
```

其成員 table 為一個指向 (Element *) 動態陣列之指標，成員 length 為此雜湊表的元素數量。雜湊表之元素會儲存於 table 動態陣列的第 (hash_function(key)) 個 index 的 (Element * 型態) 元素所指向的鏈結串列 (linked list) 中。雜湊表之元素 Element 結構定義如下:

```
typedef struct element
{
    char *key;
    int value;
    struct element *next;
} Element;
```

假設 hash function 參數為 “kiwi”、“banana”、“orange” 時，其值為 1，參數為 “apple” 時其值為 5，若元素以 (key, value) 格式表示，並依序新增元素 (kiwi, 10)、(banana, 20)、(apple, 30)、(orange, 40)，則此結構以圖像呈現為如下圖。其中，Hashtable 的成員 table 為一個動態陣列，其陣列元素為鏈結串列。鏈結串列中的元素，其成員 key 為動態字元陣列，且儲存其中之字串經過 hash function 運算後與其所在之陣列 table 之 index 相符。



本作業使用之 hash function 定義如下，其中 TABLE_SIZE 在作業模板中已經定義 (值為 100)。

```
unsigned long MurmurOAAT32 (char *key)
{
    unsigned long h = 3323198485ul;
    for (;*key;++key) {
        h ^= *key;
        h *= 0x5bd1e995;
        h ^= h >> 15;
    }
    return h % TABLE_SIZE;
}
```

請遵照上述規則實作 function:

- **Hashtable *newHashtable();**
  動態分配一個 Hashtable 結構並初始化 length 為 0，table 為長度為 TABLE_SIZE 的動態陣列，並將 table 中的元素初始化為 null。
- **void HashtableSet(Hashtable *hashtable, char *key, int value);**
  將 hashtable 中元素 key 的對應值更新為 value。若元素 key 不存在則新增元素 key。元素 key 中的鍵值為一個新分配的動態字元陣列，而不可以參考參數 key 之字串位址。
- **int HashtableGet(Hashtable *hashtable, char *key);**
  取得 hashtable 中元素 key 的對應值。若元素 key 不存在則擇一以下作法: 1) 呼叫巨集 assert，2) 回傳-9999。
- **void HashtableDelete(Hashtable *hashtable, char *key);**
  刪除 hashtable 中元素 key。若元素 key 不存在則擇一以下作法: 1) 呼叫巨集 assert，2) 不進行任何更改並 return。
- **void HashtableClear(Hashtable *hashtable);**
  將 hashtable 中所有元素刪除。
- **ItemArray HashtableGetItems(Hashtable *hashtable);**
  回傳一個可列舉 hashtable 中所有元素的結構 ItemArray (列舉之順序可隨意順序)，其成員包含結構 Item 之動態陣列 items 與整數 length 代表 items 之長度。結構 Item 成員包含動態字元陣列 key 與整數 value。將

hashtable 之所有元素之鍵與對應值不重複地列舉在 items 中。其中，每個 Item 的 key 均為新分配的動態字元陣列，而不可以參考 hashtable 內之字串位址。相關宣告如下:

```
typedef struct item
{
    char *key;
    int value;
} Item;

typedef struct itemArray
{
    Item *items;
    int length; // count of items
} ItemArray;
```

- **Hashtable *mergeHashtable(Hashtable *ht1, Hashtable *ht2);**
  動態分配一個新的 Hashtable，將 ht1 與 ht2 的內容加入其中。若有衝突的 key 則其對應值以 ht2 為準。並回傳該 Hashtable 之位址。

Scoring

請完成專案中的 simple_hash_table.c 原始碼，並僅需繳交此檔案。

此作業配分如下:

- newHashtable：10
- HashtableSet：20
- HashtableGet：10
- HashtableDelete：10
- HashtableClear：10
- HashtableGetItems：20
- mergeHashtable：10
- 良好的程式習慣：10 (包括編寫註解、程式碼縮排與有意義的變數命名)

若在上述要求有所缺陷，則會斟酌扣分。
能順利運行 main.c 不保證能夠滿分，請同學仔細檢查各功能。
若副檔名錯誤或程式有嚴重功能缺陷 (無法編譯等)，此作業不予給分。