CSE 102 Spring 2024 – Computer Programming Assignment 12

Due on May 29, 2024 at 23:59

In this assignment, you are tasked with implementing a custom dictionary data structure in the C programming language. The dictionary should provide functionality to store key-value pairs, where keys are strings, and values can be arrays of various types, including integers, floats, doubles, and characters. You must implement your dictionary in a header file named "customDict.h". You are required to have the following unions and structs:

CustomDict Structure:
The **CustomDict** structure represents the custom dictionary data structure. It contains a dynamic array of **Item** structures to store items, a capacity parameter to track current capacity of dictionary and a variable to keep track of the current size of the dictionary.

Value Union:
Implement a value union named **Value** to represent the different types of values that can be stored in the dictionary. Include members in the union to accommodate integers, floats, doubles, and characters.

Item Structure:
Implement an item structure named **Item** to represent a key-value pair in the dictionary. The **Item** structure should consist of two fields:
key: A pointer to a string representing the key.
value: A dynamic array of **Value** union representing the Value array associated with the key.

You are asked to write the following functions for the dictionary operations:

```
struct CustomDict* create_dict()
```

Description: Initializes a new custom dictionary.
Input: None.
Output: Returns a pointer to the newly created **CustomDict** structure. The capacity of dictionary should be 0 at first.

```
void add_item(struct CustomDict* dict, char* key, union Value* value)
```

Description: Adds a new item to the dictionary with the specified key and value.
Input:
- **dict**: Pointer to the custom dictionary.
- **key**: Key of the new item.
- **value**: Pointer to the value of the new item.

Output: None.

Behavior: Adds a new key-value pair to the dictionary. If the dictionary's capacity is reached, it dynamically resizes the dictionary to accommodate the new item. If there is an item in the dictionary with the given key, the function updates the existing item instead of adding new item.

```
void delete_item(struct CustomDict* dict, char* key)
```

Description: Deletes an item from the dictionary by its key.
Input:
dict: Pointer to the custom dictionary.
key: Key of the item to be deleted.
Output: None.
Behavior: Removes the item associated with the specified key from the dictionary.

```
void set_value(struct CustomDict* dict, char* key, union Value* value)
```

Description: Updates the value associated with a given key in the dictionary.
Input:
dict: Pointer to the custom dictionary.
key: Key of the item whose value is to be updated.
value: Pointer to the new value to be set.
Output: None.
Behavior: Updates the value associated with the specified key in the dictionary.

```
union Value* search_item(struct CustomDict* dict, char* key)
```

Description: Searches for an item in the dictionary by its key and returns its value.
Input:
dict: Pointer to the custom dictionary.
key: Key of the item to be searched.
Output: Returns a pointer to the value associated with the key if found, otherwise returns **NULL**.
Behavior: If the key is found in the dictionary, the function returns a pointer to the value associated with the key. Otherwise, it returns **NULL**.

```
void sort_dict(struct CustomDict* dict)
```

Description: Sorts the dictionary by keys.
Input:
dict: Pointer to the custom dictionary.
Output: None.
Behavior: Sorts the dictionary by keys in ascending order.

```
void print_dict(struct CustomDict* dict)
```

Description: Prints the contents of the dictionary in the format "Key: Value".
Input:
dict: Pointer to the custom dictionary.
Output: None.
Behavior: Iterates through the dictionary and prints each key-value pair on a new line. The entire values in the value array of an item should be printed on a single line.

```
void free_dict(struct CustomDict* dict)
```

Description: Frees the memory allocated for the dictionary.

Input:
dict: Pointer to the custom dictionary.
Output: None.
Behavior: Deallocates the memory used by the dictionary and all its elements.

```
int read_csv(struct CustomDict* dict, const char* filename)
```

Description: Reads data from a CSV file and populates the dictionary accordingly.
Input:
dict: Pointer to an empty custom dictionary.
filename: Name of the CSV file to be read.
Output: Returns **1** on success, **0** on failure.
Behavior: Reads data from the specified CSV file and populates the dictionary with key-value pairs. Each row of the CSV file represents a key-value pair, where the first column indicates the data type (int, float, double, char), second column is the key and subsequent columns represent the values.

Use the read_csv() function to read the provided data.csv file and populate your dictionary. Write a main.c file to include and utilize the custom dictionary. Ensure that all requested functionalities are tested in the demo video and explicitly explained in the report file.

IMPORTANT NOTES:

- Submit your homework as a zip file named as your student id (StudentID.zip) and this file should include:
  - YourStudentID.c file and customDict.h file
  - A pdf file named "YourStudentID.pdf" including a YouTube link and screenshots of your program outputs. In the video, you are expected to provide a demo of your assignment. For each requested functionality, you must explicitly explain your solution approach and also execute and display the outputs. The video should not exceed 4 minutes. Please ensure that your camera is turned on during the recording.
- Compile your work with given command “gcc --ansi your_program.c -o your_program”.
- Your work will be evaluated using gcc version 11.4.0.
- For any questions and problems, you can always contact me **via email** (b.koca@gtu.edu.tr), or you can find me in Room 119 during scheduled office hours on April 21, 2024, between 13:30 and 14:30.