PA301 – Simple Translation Dictionary

We want to implement a program which will help students learn the vocabulary of foreign languages. The first step, which has been assigned to you, is to build a data structure representing a list of English words and their translation in a foreign language.

We'll also need to provide functions allowing us to lookup translations from English-to-Foreign and Foreign-to-English. The main data structure of this program encapsulates some variables specifying the maximal and current number of words in the dictionary along with a pointer to a dynamically allocated array of pointers on another data structure representing a pair of words.

A skeleton for this project is provided to you along with the struct definitions.

Files you will be working with

You will be provided with several files to get you started working on this assignment.

- You must not alter the file names, remove or add files to the project
- You must only modify the ones marked below with a yes in "Modify it?"
- You must not insert any comments or code in the *tests.c* file which, when read by another student, would give them any insights about the solutions you implemented in *tools.c*.

Important Academic Honesty Note;

> *The role of tests.c is to allow you to test your program to verify it adheres to requirements. Your instructor might allow you to exchange this file, and this file only, with other students. Therefore, you must uphold academic honesty standards by not inserting any information, besides the tests, which would divulge your design or implementation of the solutions to another student. Failure to do so will earn you a FF for the offering.*

Here are the files;

File name	Modify it?	Role
tools.c	**Yes**	Implementation of your solution to the assignment
tests.c	**Yes**	Implementation of your test functions
tools.h	**No**	Header file for tools.c
main.c	**No**	Implementation of the main function starting your tests
testlib.h	**No**	Definition of the TEST function you must use in your tests
testlib.c	**No**	Implementation of the above

Task #1 – Implement and test dictionary_build

This first function, which you will implement in *tools.c*, will be used every time you need to start working on a new dictionary. Part of your grade on this implementation will be on how well your function handles error situations; e.g. we ask for a dictionary of negative size, there is no memory available for your *malloc*…

Any time you encounter one of these, or similar, errors, you should return a specific value from your function which is described in the program's source itself. When you test your function with such parameters, you should verify its return value is appropriate.

Make sure you follow our usual process when working on this function;

- Read the requirements & existing source thoroughly
- Implement your tests based on how you understand the requirements. This time you will note that we are not providing tests at all. Part of your work is to leverage what you learned in the previous PAs to design & implement your own tests
- Implement the function dictionary_build
- Update your tests based on what you learned
- Refactor your solution before to go to the next task.

Task #2 – Implement and test dictionary_free

This 2nd function will be used to de-allocate a dictionary. Here are some things to keep in mind;

- Make sure you dispose of all the memory you allocated in a struct dictionary. Drawing a diagram of the data structures will help you figure out what you are de-allocating as you write your *dictionary_free* function. This should help you avoid memory leaks.
- If the dictionary passed to your function is only partially allocated – e.g. due to a bug elsewhere – you should handle it gracefully; i.e. before to free a pointer, make sure it is not NULL.

Follow the steps outlined in task#1 as you work on implementing this task. Again, the source code provides you with most of the information you need in the form of comments and the code itself.

Task #3 – Implement and test dictionary_add

The 3rd function will take a dictionary and two strings as parameters. It will use **strdup** to make a copy of the words provided as parameters and insert them as a new element in the dictionary.

Again, detect possible bad parameters; e.g. no words provided; 1 word instead of 2… and verify the return value is appropriate based on the documentation in the program itself.

Follow the steps outlined in task#1 as you work on implementing this task.

Task #4 – Implement and test dictionary_translate

This function, as indicated in its comments, will return a **strdup** copy of the translation of one of the two strings passed as parameter. It is expected that one will use this function with a valid dictionary, and specify one of the two strings to be NULL.

- If the English word parameter is NULL, we are providing a string in the foreign language to be translated into English.
- If the foreign word is NULL, we do the opposite.

Passing two non-NULL strings or a NULL dictionary are errors which should be handled by your function returning the value described in the documentation in the program itself.

Follow the steps outlined in task#1 as you work on implementing this task.