Politically correct texts

Submission deadline:	**2022-12-05 11:59:59**	833310.555 sec
Late submission with malus:	**2022-12-31 23:59:59** (Late submission malus: 100.0000 %)	
Evaluation:	**1.1000**	
Max. assessment:	**4.0000** (Without bonus points)	
Submissions:	7 / 20 Free retries + 10 Penalized retries (-10 % penalty each retry)	
Advices:	2 / 2 Advices for free + 2 Advices with a penalty (-10 % penalty each advice)	

Your task is to develop a function (not a whole program, just a function) which helps PR departments formulate politically correct texts.

Political correctness is a movement leading to changes in our language use. The target is to formulate texts such that the text does not offend anybody. The term "politically correct" is official, people sometimes use "newspeak" to refer to this form. The most politically incorrect people even claim there may be some parallels between the political correctness and Orwell's novels. Anyway, it is unacceptable to use the word "slacker" when referring to a person. Instead, "person with a specific need for workload planning" is correct.

Your implemented function will help the spread of political correctness. It is tedious to use the long forms of politically correct phrases, even for people without any specific needs for workload planning. Your function will address the issue, it will simply convert the politically incorrect text into the acceptable output. The interface will be:

```
char * newSpeak ( const char * text, const char * (*replace)[2] );
```

The parameters are:

- `text` - the source text where the function shall replace some phrases. The parameter is `const`, thus the function shall only read it,
- `replace` - is a 2D array of string pairs. There are `n` rows and 2 columns in the array. The number of rows corresponds to the number of phrases in the replacement dictionary. Column 0 stands for the politically incorrect string, column 1 is the replacement. The last row in the array contains two `NULL` pointer.
- Return value is a string with the replacements applied. The string must be dynamically allocated in the functions (using either `malloc` or `realloc`). The caller will use the string and will free it using `free` when the string is no longer needed. If the parameters are incorrect (see below), the return value must be `NULL`.

We require unambiguous operation. To avoid any ambiguity, we adopt the following:

- text is replaced from left to the right,
- we never return back to the text already replaced. Instead, we continue after the replacement. This is demonstrated in sample runs with word "specialist".
- the replacement is unique. In other words, no string in column 0 is a prefix of any other string in column 0. The function shall tests this restriction before it starts replacing. If the restriction is viloated (i.e., some string in column 0 is a prefix of some other string in column 0), the function returns `NULL` as an error signaling. This is demonstrated in the sample run below, pair `fail` and `failure`.

All string are C strings (zero terminated strings). All string comparisons are case sensitive.

Submit a source file with the implementation of the required function `newSpeak`. Further, the source file must include your auxiliary functions which are called from `newSpeak`. The function will be called from the testing environment, thus, it is important to adhere to the required interface. Use the attached sample code as a basis for your development, complete `newSpeak` and add your required auxiliary functions. There is an example `main` with some test in the sample below. These values will be used in the basic test. Please note the header files as well as `main` is nested in a conditional compile block (`#ifdef/#endif`). Please keep these conditional compile block in place. They are present to simplify the development. When compiling on your computer, the headers and `main` will be present as usual. On the other hand, the header and `main` will "disappear" when compiled by Progtest. Thus, your testing `main` will not interfere with the testing environment's `main`.