R5: Dynamic Arrays & Strings

General Instructions:

1- Create a C project in eclipse called R5. Extract the file R5.zip into a folder of your choice at your machine. Copy the files into the R5 project.

2- Rename the file: “R5_template.txt” to “R5.c”. Enter your credentials on top of the file.

3- Follow the videos posted by the instructor on how to complete the tasks presented in this worksheet.

4- To test your code, select the proper function in the file: “R5_test.c”. Compare your results with those presented in “R5_output.txt”. Your results should exactly match the output.

5- Submit ONLY R5.c file.

Task 1 Creating Dynamic Arrays

Implement the following two functions:

1- `int* get_array1(int start, int end);`

2- `long* get_array2(long start, long end);`

The functions create a dynamic array of type int for the first function and of type long for the second function. The array contains elements containing values from start to end, inclusive both ends.

In the first function uses *malloc*, and the second uses *calloc* to create the arrays dynamically. A pointer to the newly created array is returned by each function.

Task 2 Updating Dynamic Arrays:

Implement the following two functions:

1- `void insert_item(int *array, const int size, int item, int pos)`
2- `void remove_item(int *array, const int size, int value)`

Both functions receive a pointer to a dynamic array along with tis size. The first function inserts given value at the specified position, and the second removes the given value.

In both functions, if the given pointer is NULL, the function prints an error message and returns. In the first function, if the given position is invalid, an error message is printed. In the second function, if the given value is not found, an error message is printed.

Both functions should use *realloc*, to modify the array size after performing the insertion/removal operation.

Task 3 Initializing a String:

Implement the function: `void init_str();`

Follow the instructions in the videos which present five different methods to initialize a string:

1- Using fixed size array and strcpy function.
2- Using fixed size array and array initialization
3- Using dynamic memory allocation of a character array block
4- Using assignment of a string to a character pointer
5- Using assignment of a string to a character array

Task 4 Reversing a String:

Implement the following four functions which reverse a string using different methods:

1- `void reverse_str1(char *s)`: Reverses a string using built-in functions in the string.h library.
2- `void reverse_str2(char *s)`: Reverses a string manually and in-place.
3- `char* reverse_str3(char *s)`: Reverses a string manually and in-place but also returns a pointer to the string
4- `char* reverse_str4(char *s)`: Keeps the current string unchanged but creates a reverse string and return a pointer.

Task 5 Swapping Strings:

Implement the following two functions which swap two given strings:

1- `void swap_str1(char *s1, char *s2)`: Swaps two strings by changing their values using copy by reference.
2- `void swap_str2(char **s1, char **s2)`: Swaps two strings by changing their pointers.

Task 6 Has Double?:

Implement the function: `int has_double(char *s)`

The function returns true if given string has any two (or more) similar characters, regardless of their position in the string, i.e. it is indifferent to the two characters being contiguous or not. The function works only on alphabetical characters and ignores all other characters. The function is not case sensitive, i.e. lower and upper case characters are treated the same.

Task 7: Additional Task:

Follow the instructions provided by the instructor regarding the additional task of this week. The task will be presented during the Wednesday lecture.

Qutaiba Albluwi © 2021