/\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

\* Program: Lab 30 Part C - Comp 183 \*

\* Author: Matthew Casiro \*

\* Submitted: Feb 11 2016 @ 12:29:59 \*

\* Purpose: Given a filter character set and string character set, \*

\* return the string if all characters appear in the filter \*

\* string. Otherwise, return NULL and set an error message. \*

\*\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*/

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

#define INITIAL\_SIZE 10

#define NUM\_CHARS 256

// Take an input array and the size of that array. Create

// a char array of double the original size and copy over all

// values to the new array. Return NULL if any issues,

// or a pointer to the new array if successful.

**char \*doubleArraySize(char \*oldArray, unsigned int \*size)**

{

unsigned i, newSize;

// If Array pointer, Array, Size pointer, or Size are

// NULL (or 0), then return NULL.

if (!oldArray || !\*oldArray ||

!size || !\*size) {

return NULL;

}

newSize = (\*size) \* 2;

// Allocate a new array of double the size, if calloc fails

// print error and exit program.

char \*newArray = calloc(newSize, sizeof(char));

if (!newArray) {

printf("FATAL ERROR: Calloc failed. Exiting Program.\n");

exit(1);

}

// Copy values from original array to new array

for (i = 0; i < \*size; i++) {

newArray[i] = oldArray[i];

}

\*size = newSize;

return newArray;

}

// Calculate and return the length of a zero-terminated string.

**unsigned stringLength(char \*string) {**

unsigned length = 0;

// If string pointer, or string is NULL or zero, return 0

if(!string || !\*string) {

return 0;

}

while (string[length] != 0) {

length++;

}

return length;

}

// Build a boolean filter array indexed to correspond to an ASCII table,

// prompt a user for an input string, and output the string if all

// chars in the string are present in the filter string.

**char \*getFilteredString(char \*prompt, char \*filter, char \*\*errorMessage) {**

unsigned i = 0, j = 0, size = INITIAL\_SIZE;

char input;

\_Bool filterTable[NUM\_CHARS];

char \*tmp = NULL;

char \*output = calloc(INITIAL\_SIZE, sizeof(char));

// Initialize boolean array to false, then iterate through

// filter string and flip values corresponding to ASCII key

// for each character to true

for (i = 0; i < NUM\_CHARS; i++) {

filterTable[i] = 0;

}

i = 0;

if(filter && \*filter) {

while (true) {

input = filter[i];

if (input == '\n' || input == EOF) {

break;

}

filterTable[input] = 1;

i++;

}

}

// Prompt user for input if pointer to prompt string is valid

// and points to a valid string

if (prompt && \*prompt) {

printf("%s", prompt);

}

input = getc(stdin);

// Check input buffer against filter table

while (input !='\n' && input != EOF) {

// If the array holding the input string is full,

// request a new array of double the size

if (j == size) {

tmp = doubleArraySize(output, &size);

output = tmp;

free(tmp);

}

// If currect character is 'true' in the filter table,

// add that character to the output string

// Otherwise set an error message and exit

if (filterTable[input]) {

output[j] = input;

} else {

if(errorMessage) {

\*errorMessage = "Found invalid character";

}

return NULL;

}

input = getc(stdin);

j++;

}

return output;

}