CSE 1320: Intermediate Programming
University of Texas at Arlington
Spring 2021
Dr. Alex Dillhoff

Assignment 3

1. Create a function `swap_strings` which takes two arrays of `char *`, the sizes for each array as `int`, and two index values. The function should swap the string in the first array at the first index with the string at the second index in the second array.

   In main, initialze two arrays of `char *` with at least 4 strings using inline initialization (`char *arr[] = { ... }`). Print out the strings in each array and prompt the user to enter 2 index values. Swap the strings with the `swap_strings` function and print the resulting arrays after the swap.

   - You may only use `stdio.h`.
   - Save your code as `swap_strings.c`.

2. Create a function `dot_product` which accepts two pointers-to-double as input along with an integer reflecting the length of both arrays (they are the same size). Using pointer arithmetic to access the array elements, compute the dot product of the two arrays. The function should return `double`.

   In main, read in two arrays of 8 values from the user following the I/O format seen in the example run below. On a new line, print the output following the format shown below.

   The dot product is defined as

   $$\mathbf{a} \cdot \mathbf{b} = \Sigma_{i=1}^{n} a_i b_i$$

   Save your code as `dot_product.c`.

   Example Run

   ```
   > -1 8 6 9 3 -9 7 8
   > 3 5 5 -2 3 -6 4 -9
   68
   ```

3. Implement a substring search and reverse function which searches a string for a given keyword. The function should return a pointer to the address of the last character of the substring within the original string, if it exists. Additionally, the substring should be modified so that it is reversed. If the substring is not found within the string, return `NULL`.

If the substring is found and reversed, print the modified string. If no substring is found, print `"Key not found!"`

Your program's input and output should match what is seen in the example run.

- You may only use `stdio.h`.
- Save your code as `swap_and_reverse.c`.

Example Run

```
Enter string: this is a test string.
Enter key: test
this is a tset string.
```

4. Write a program that accepts an input string from the user and converts it into an array of words using an array of pointers. Each pointer in the array should point to the location of the first letter of each word.

   Implement this conversion in a function `str_to_word` which returns an integer reflecting the number of words in the original string. To help isolate each word in the sentence, convert the spaces to `NULL` characters.

   You can assume the input string has 1 space between each word and starts with an alphanumeric character.

   Your program's input and output format should match the example run below.

   - You may only use `stdio.h`
   - Save your code as `str_to_word.c`.

   Example Run

   ```
   > This is a string.
   4 word(s) found.
   ```

5. A social networking company, Chirper, has contracted you to parse their CSV data of individual Chirps. Each line of CSV in their file is structured as "USERNAME,MESSAGE,DATE".

   Implement a function `parse_chirp` which verifies that each input string matches this format by ensuring that:

   - There are 3 tokens in the string corresponding to the 3 properties above.
   - The username must begin with `@`.
   - The third token should be in the format `MM/DD/YYYY` (hint: tokenize this as well).

   The function should return 0 if the CSV string is valid, and 1 otherwise.

   Your program should read CSV strings until the string `"END"` is read as input. Print all valid strings to `stdout` followed by a line indicating how many invalid strings were entered.

- Do not use file operations for this assignment.
- **Hint:** A continuous stream of strings can be entered by constructing a CSV file using the properties defined above and putting each entry (USERNAME,MESSAGE,DATE) on a newline. You can then redirect the contents of the file into your program (`./a.out < file.csv`)
- You can use a character array with a buffer size of 1024 for reading each line using `fgets`.
- Each chirp can be no longer than 140 characters.
- You may not use any library specifically for parsing CSV.
- Test your code using the file `chirp_data.csv`, available through Canvas. You should be able to input it to your program using redirection.
- Save your code as `parse_chirps.c`.

Create a `zip` file using the name template `LASTNAME_MAVID_A3.zip` which includes the all required code files. Submit the `zip` file through Canvas.