CS/240/Lab/3

Q1: Words

In this lab, we will complete the parser by implementing the functions in `libwords.a`. The specifications are given below. Download `words.c` from Piazza and complete the function stubs. Don't forget that you have `utils` lib functions available to use.

`int readMsg(char* buf)`

Description: Read one line from standard input into `buf`. (`\n` will be put in `buf` too if there is one.) Return the length of the message. Return EOF if there are no more lines to read. This function does not null-terminate the string.

`int getWord(char* msg, int len, int* start, int* end)`

Description: Find the next word. `msg` is a pointer to an array of characters. `len` is the length of the `msg` array. `end` is a pointer to an integer that holds the current position in the array of characters. The function will find the next word, starting from the position given by the initial value of `end`.
If a word was found the function returns 1, and updates `start` to point to the index of the first character in the word, `end` to point to the index of the first character after the word.
If no word was found, the function returns 0.

`void checkWord(char* word, int len, char** keywords, int num)`

Description: Check if `word` is contained in `keywords`. If yes, erase `word` with white spaces. `len` indicates the length of `word` and `num` indicates the length of the keyword list.

`void unCapitalize(char* c)`

Description: Change the character pointed by `c` to lower case if it is a capital.

Compiling & Testing

You will receive the reference implementation `parser_ref.o` and `libutils_ref.a` for building the parser. But before putting everything together, do unit testing first and make sure your `words.c` satisfy the autograder. The command to create `libwords.a` and link it with the other two components is

```
gcc -std=c99 -c words.c
ar rcu libwords.a words.o
gcc -std=c99 -o parser parser_ref.o -L. -lwords -lutils_ref
```

To test with messages in `msg.txt` and keywords 'in', 'of' and 'the', use

```
cat msg | ./parser in of the
```

As a reference, the specifications of the utils functions are repeated below.

`int isAlpha(char ch)`

Description: Test if `ch` is a letter (a - z or A - Z). Return 1 if yes, otherwise 0.

`int strLen(char* s)`

Description: Return the length of the `'0'`-terminated string `s`. The `\0` character is not counted, e.g. the length of `"abc"` is 3.

`int strNCmp(char* s1, char* s2, int n)`

Description: Compare the first `n` characters of string `s1` and `s2`. Return 1 if identical, 0 otherwise.

`void spaceOut(char* s, int len)`

Description: Replace every character of `s` with a white space. `len` indicates the length of `s`.

Turning in

Go to the course web page (`http://web.me.com/vitekj/240s12/`), and under the "submissions" section click "web site". On the submission web page, enter your unique turnin ID in the box and press "log in". Under "currently open projects" will be listed "lab 3 words". Click "lab 3 words"', use the file selector to choose your `words.c` file, then click "submit". The page will tell you how you've done, and your total score. When you're satisfied, close your browser window. You may resubmit at any time before the due date by the same process.

Lab is due Mon., Feb. 6th before midnight. No late labs accepted.

Grading criteria

Words

- source file is named `words.c`
- code compiles
- code runs without error
- code only includes `stdio.h`, `utils.h`, and `words.h`
- `readMsg()` - It must read the whole line including the last `\n` if there is one and return the correct length. It must return EOF when getting to the end of file and the current line is empty.
- `getWord()` - It must respond properly on empty string, strings that contain or don't contain words, strings that contain any arbitrary ASCII characters.
- `checkWord()` - It must respond properly on zero length words, zero length keyword list, words contained or not contained in keyword list.
- `unCapitalize()` - It must be able to turn all uppercase letters into lowercase while leave the rest ASCII characters untouched.

[Optional] Compiling & Testing with own code from Lab2

You could also try testing your code with your code from Lab2. First copy over parser.c and utils.c to the current directory. You can then compile it as follows:

```
gcc -std=c99 -o parser parser.c words.c utils.c
```

To test with messages in `msg.txt` and keywords from `keys.txt` (which you must create), use

```
cat msg.txt | ./parser $(cat keys.txt)
```