CS 2022 – Fall 2011
Assignment #3
9/19/2011
Due: Saturday 9/25/2011 11:59 PM

In this assignment you are asked to write a small interactive C program to demonstrate command of I/O and other principles we have discussed in class so far.

Interactive Manipulator

You are asked to write a C program `iManip.c` that will interact with the end-user, accept keyboard input during execution, and output the results based on the user's commands. More specifically, your program will support the following "*commands*" typed-in by the end-user during runtime:

- **Bit operations:**
  - `bit_or <integer1> <integer2>` - bitwise OR of the two specified integers.
  - `bit_and <integer1> <integer2>` - bitwise AND of the two specified integers.
  - `bit_xor <integer1> <integer2>` - bitwise XOR of the two specified integers.
  - `bit_shift_left <n> <k>` - shift the bits of the integer `n` `k` bits to the left.
  - `bit_shift_right <n> <k>` - shift the bits of integer `n` `k` bits to the right.
- **String Operations:**
  - `reverse <string>` - reverse of the specified string.
  - `caesar_cipher <string> <key>` - shift every letter of the string "*forward*" by the number of letters specified by the key. For instance, if the key is 2 and the string is "Hello", the resulting string is "Jgnnq", since 'j' is two letters in the alphabet after 'h', 'g' is two letters in the alphabet after 'e', etc.

    This caesar cipher should "wrap around" once the end of the alphabet is reached. For instance, if the key is 3, 'z' should be shifted to 'c': shifting 'z' once yields 'a', shifting 'a' yields 'b', and shifting 'b' the third time yields 'c'. As another example, the string "Zany" with key 4 yields "Derc".

    The caesar cipher should preserve the case of alphabetic characters, as the previous examples showed. Finally, you can expect that the caesar command will only be called with alphanumeric input (i.e. strings that are made of letters and numbers only, no special characters).
- **Batch Processing:**
  - `process_file <input file name> <output file name>` - read the specified input file, process the commands in it, and output the results to the specified output file. You should create the output file if it does not exist. You should also print an error message if the input file does not exist.

    The input file can contain any list of the commands mentioned above along with their inputs. Each command and its arguments will appear on a separate line.

The output file shall have the result of executing each command on a separate line. The results should be in the same order as their corresponding commands appear in the input file.

- **Termination:**
  - `exit` - stop the execution of your program.

When your program is started, it should ask the user for a command to run. Once the user types-in the command, your program should do the processing and print the results to the screen (or to a file in the case of `process_file`), then it should ask the user for another command to run. This shall continue until the user types the `exit` command, at which point your program terminates.

Here is an example run:

```
:~> ./iManip
>> Please enter a command to run.
bit_or 15 25
31
>> Please enter a command to run.
bit_and 15 25
9
>> Please enter a command to run.
bit_xor 15 25
22
>> Please enter a command to run.
reverse hello
olleh
>> Please enter a command to run.
reverse CS_2022_Intro_to_C
C_ot_ortnI_2202_SC
>> Please enter a command to run.
caesar_cipher HelLo 3
JgnNq
>> Please enter a command to run.
caesar_cipher Zebra2009 1
Afcsb3110
>> Please enter a command to run.
caesar_cipher Zebra2009 5
Ejgwf7554
>> Please enter a command to run.
process_file nonexistent_file.txt output.txt
Error! Input file "nonexistent_file.txt" does not exist!
>> Please enter a command to run.
process_file input.txt output.txt
Done processing. Output written to "output.txt".
>> Please enter a command to run.
exit
Bye Bye!
```

For file processing, if the input file was the following:

- ```
  bit_or 15 25
  bit_and 15 25
  bit_xor 15 25
  ```
- ```
  reverse hello
  reverse CS_2022_Intro_to_C
  caesar_cipher HelLo 3
  caesar_cipher Zebra2009 1
  caesar_cipher Zebra2009 5
  ```

The output file should have the following content:

- ```
  31
  9
  22
  olleh
  C_ot_ortnI_2202_SC
  JgnNq
  Afcsb3110
  Ejgwf7554
  ```

To simplify things, assume that input files do not allow the command "`process_file`" in them. That is, files can not issue commands to process other fies.

Tips

- Use the function `atoi`, defined in `stdlib.h`, to convert string input from the user to an integer for the bitwise operations.
- `char`s have numerical values in ASCII (for example `'a'` has the numerical value `97`, `'A'` has value `65`, and `'0'` has value `48` in ASCII). Additionally, sequential characters have sequential numerical values (for example, `'b'` has value `98` and `'c'` has value `99`). Thus, to get the character forward from `char c`, you can just do `c+1`.
- Use the modulo operator (`%`) to wrap around alphanumeric ASCII characters for the caesar cipher.

Submission and Testing

Submit your work on CMS (http://cms.csuglab.cornell.edu/) by the deadline. Make sure you have been added to CMS early on and otherwise contact me via email. You should submit the **source code** (the .c files) of your application and **not** the compiled binaries.

Your programs will be compiled with `gcc` and tested on a linux environment. You are free to choose the environment of your liking to develop your solutions, but keep in mind that testing will be on a fixed environment, and your application is expected to run on that.

Academic Integrity Reminder

Remember that you may have general discussions about how to approach this problem with your peers, but you should work on the final solution by yourself alone. If you are stuck or are having trouble, you may email me or talk to me after class on Monday or during my office hour
  ```

on Wednesday.

Good Luck!