CS2640 Project 4
Encryption Utility
Total points: 100

Project Purpose (important):

The purpose of this project is to test your knowledge of subroutines, reading and writing to I/O devices, specifically , the console, reading strings from user input, as well as reading and writing files.

Project Specifications:

Write a program in mips assembly language that prompts the user for a source file path, a destination file path, and a passphrase.

To read the file paths from the console, you will prompt the user for the path and then use syscall to collect the string from console. **The buffers that hold these path strings must be 256 bytes large**. This must be done using a subroutine.

To collect the passphrase you must read each character that is input at the console. As you read these characters you will output an asterisk character to console to hide the passphrase. The collection of the passphrase should be a separate subroutine. You must read each character from the receiver (known as polling), Store that character into a buffer, then send an asterisk character to the console (transmitter) and loop around to capture the next passphrase character. You will break out of this Loop when the enter key is pressed and store and null character at the end of the passphrase in the buffer.**Your buffer must be 257 bytes** (able to store a 256 character passphrase).

The program then reads a buffer load of data at a time from the source file and performs an XOR on each byte of the buffer and the passphrase. The resulting bytes are stored in the same buffer overwriting the byte that was just read, then the buffer gets written to the destination file. **The buffer used for this process must be 1024 bytes in size.** It is important that you allow for any size source file. You should test with plain text source files that are larger than 1kB in size. The resulting encrypted file should be exactly the same size as the source file. To decrypt the encrypted file, run your program again and enter the path to the encrypted file as the source file path. You can assume the passphrase will be less than or equal to 256 characters (plus a null terminator). This process must be implemented as a subroutine,

Design your subroutine to take as input parameters, the source path, the destination path, and the passphrase. The parameters will hold the addresses to the buffers that hold the strings.

To get full credit, minimize the size of the main routine, in other words, create small, testable subroutines just as you would when programming in a high-level language.

Refer to the document called, “syscall codes”, posted in the resource section on Blackboard for information on how to read strings and read/write files. Also refer to the lecture notes posted on blackboard as well as your class notes.

Be sure to Include pre and post condition comments for your subroutines and comment your code thoroughly.

Your assembly program must run in the MARS simulator.

important: To get full credit for this project, you must make sure to ignore the null termination characters at the end of the passphrase for reasons stated in our lecture. There will be very little partial credit given for this project. Please read the project carefully and be sure to create the structure that is expected of you. If you are not sure about any part of this project, be sure to ask.

Submission Instruction:

Submit your work in this manner:

- Copy your files to a **folder** named, “your_name_p4” where “your_name” is your first and last name separated by an underscore. Compress that folder into a single Zip archive which should be named, “your_name_p4.zip”
- Submit your zip file via Blackboard

(NO LATE SUBMISSIONS ACCEPTED FOR ANY REASON)

The algorithm for this project:

```
/*
     getString uses syscall to read a string into a buffer from console.
     this buffer should be able to contain at least 256 characters. The
     buffer must be null terminated.
/*
src_filepath = getString(src_buffer)
dst_filepath = getString(dst_buffer)

/*
     getPassphrase uses I/O Programming to read each char of the
     passphrase from the keyboard, store it in a buffer (The buffer must
     be 257 bytes), and echo an asterisk to the console. This passphrase
     is finished when the user presses the enter key. This buffer must be
     null terminated. You must ensure that the user inpute less than 257
     characters.
*/
passphrase = getPassphrase(pass_buffer)

/*
     This subroutine opens the source file in read mode, and the
     destination file in write mode

     It loops through the source file, reading a buffer full of data at a
     time. it loops over the buffer (hint, the last buffer that is read
     only be partially full) and XOR each byte of the buffer with a byte
     in the passphrase storing the resulting byte back to the buffer. The
     buffer is then written to the destination file. The passphrase will
     most likely be much smaller than the buffer so when the end of the
     passphrase is reach in the XOR loop, start again at the beginning of
     the passphrase; do not XOR buffer data with the null terminator at
     the end of the passphrase. Between buffer loads, do not lose your
     place in the passphrase, you should start XORing at the character in
     the passphrase where you left off from the previous buffer load.

     The flag codes required for opening a file for read or write are:
     flags: 0 = read-only
            1 = write-only with create
            9 = write-only with create and append

     mode:  0 = ignored.
*/
encryptFile(src_path_buffer, dst_path_buffer, passphrase_buffer)
```