Program Submission Instructions:

- You must submit your source code file
- The source code file must be submitted in Webcourses from the assignment page
- All source code must be in exactly one file of type .c, .cpp, or .java

CIS 3360 – Security in Computing
Spring 2021
HW #2 Checksum (100 points)

In this assignment you'll write a program that calculates the checksum for the text in a file. Your program will take two command line parameters. The first parameter will be the name of the input file for calculating the checksum. The second parameter will be for the size of the checksum (8, 16, or 32 bits). The program must generate output to the console (terminal) screen as specified below.

Command Line Parameters

1. Your program **must** compile and run from the command line.
2. The program executable must be named "checksum" (all lower case, no spaces or file extension).
3. Input the required file names as command line parameters. Your program may NOT prompt the user to enter the file names. The **first parameter** must be the name of the file used for calculating the checksum, as described below. The **second parameter** must be the size, in bits, of the checksum. The sample run command near the end of this document contains an example of how the parameters will be entered.
4. Your program should open the two files, echo the processed input to the screen, make the necessary calculations, and then output the ciphertext to the console (terminal) screen in the format described below.

Checksum size

The checksum size is a single integer, passed as the second command line argument. The valid values are the size of the checksum, which can be 8, 16, or 32 bits. Therefore, if the second parameter is not one of the valid values, the program should advise the user that the value is incorrect with a message formatted as shown below:

```
printf("Valid checksum sizes are 8, 16, or 32\n");
```

The message should be sent to STDERR.

Format of the input file

The input file will consist of the valid ASCII characters associated with the average text file. This includes punctuation, numbers, special characters, and whitespace.

Output Format

The program must output the following to the console (terminal) screen:

1. Echo the input file
2. Print the checksum.

The echoed input text should be in rows of exactly 80 characters per row (not counting the NEWLINE character), except for the last row, which may possibly have fewer. These characters should correspond to the input text. The checksum line should be formatted as follows:

```
X bit checksum is Y for all ZZZ chars
```

Using the following:

```
printf("%2d bit checksum is %8lx for all %4d chars\n",
checkSumSize, checksum, characterCount);
```

Where X is the checksum size (*checkSumSize*) of 8, 16, or 32, Y is the calculated checksum (*checksum*), and ZZZ is the character count (*characterCount*) of the input file. Note that the checksums are **masked** to print the appropriate sizes such as two hex characters for 8 bit checksum, 4 hex characters for the 16 bit checksum, and 8 hex characters for 32 bit checksum. *(Note that the format specifier for the checksum is %8lx, or the percent sign, the number 8, the letter l, and the letter x.)*

What to submit to WebCourses

You must submit this assignment's source code, appropriately commented, via WebCourses.

Program Notes and Hints

- Your program must read in an input text file that may contain uppercase letters, lowercase letters and non-letter characters. Your program should then calculate the checksum

appropriately for the size specified in the command line. Specifically, if the checksum is 8 bits long, each character should be used as the number to be added to the checksum. Likewise, if the checksum is 16 bits long, each two characters should be added to the checksum.

- Note that there is a 50% chance that there will be one character short on the input file. In that case use the character "X" (an uppercase X) as the pad character. Similarly, if the checksum is 32 bits, use the same technique and character to pad the input string appropriately with 1, 2, or 3 pad characters (*X*).
- Correspondingly, consideration of the file termination character NEWLINE will be required to match the expected output files. This is due to the fact that each input file is terminated by a NEWLINE character, the hexadecimal '0A'.

- Note that this assignment does NOT behave the same as explained in the lecture. There is NO REQUIREMENT to calculate the two's complement of the checksum total.

Sample Run Command

C or C++ program:

```
Prompt$ ./checksum  inputText1.txt 8
```

Java program:

```
Prompt$ java checksum inputText1.txt 8
```

Source code comment blocks (REQUIRED)

The **header** of the source code file should contain the following comment block:

```
/*============================================================================
 |  Assignment:  HW 02 – Calculating the 8, 16, or 32 bit checksum for a
 |                    given input file
 |
 |   Author:  Your name here
 |   Language:  Java
 |
 |  To Compile:  javac Hw02.java
 |
 |  To Execute:  java Hw02 textfile.txt checksum_size
 |                    where the files in the command line are in the current directory.
 |
 |                   The text file contains text is mixed case with spaces, punctuation,
 |                   and is terminated by the hexadecimal '0A', inclusive.
 |                   (The 0x'0A' is included in the checksum calculation.)
 |
 |                    The checksum_size contains digit(s) expressing the checksum size
 |                    of either 8, 16, or 32 bits
 |
 |  Class:  CIS3360 - Security in Computing – Spring 2021
 |  Instructor:  McAlpin
 |  Due Date:  per assignment
 |
 +============================================================================*/
```

The last lines of your code should contain the following **Academic Integrity** statement:

```
/*============================================================================
 |    I [your name] ([your NID]) affirm that this program is
 | entirely my own work and that I have neither developed my code together with
 | any another person, nor copied any code from any other person, nor permitted
 | my code to be copied  or otherwise used by any other person, nor have I
 | copied, modified, or otherwise used programs created by others. I acknowledge
 | that any violation of the above terms will be treated as academic dishonesty.
 +============================================================================*/
```

Grading Rubric

The total possible score for this program is 100 points. The following point values will be deducted for the reasons stated:

[-100 points] Your program does not successfully compile from the command line with one of these commands:

C program:	prompt>	gcc –o checksum checksum.c
C++ program:	prompt>	g++ –o checksum checksum.cpp
Java program:	prompt>	javac checksum.java

Note: If you are submitting a Java program, the class file must be named "checksum.java" and the class name must be "checksum".

[-100 point] The program does not accept input file names from the command line.

[-90 points] Your program does not run from the command line without error or produces no output.

[-70 points] The program compiles, runs, and outputs the input file (with a maximum of 80 characters per line), but crashes thereafter or produces no checksum output.

[-25 points] The program compiles, runs, echoes the input, and calculates the checksum, but the checksum output is incorrect and it is not formatted correctly (not all input text or not a maximum of 80 characters per line).

[-10 points] The program does not calculate the checksum correctly for the specified checksum value. Note that in the event that more than ONE checksum calculation is incorrect, a 10 point deduction will be applied. For example, if the 8 bit checksum passes but the 16 and 32 bit checksums are incorrect, 20 points will be deduction; 10 for 16 bits, and 10 for 32 bits.

[no deductions] The program compiles, runs, echoes the input (80 letters per line) and calculates the appropriate checksum value.

Test Files

The following test files are included for your program testing. All of them are in the ZIP file names hw2Testing.zip on Webcourses.

hw2Test.sh	s10A-Output32.txt	s18A-Output8.txt
in10A.txt	s10A-Output8.txt	sRF2-Base16.txt
in17A.txt	s17A-Base16.txt	sRF2-Base32.txt
in18A.txt	s17A-Base32.txt	sRF2-Base8.txt
in19A.txt	s17A-Base8.txt	sRF2-Output16.txt
inRF1.txt	s17A-Output16.txt	sRF2-Output32.txt
inRF2.txt	s17A-Output32.txt	sRF2-Output8.txt
inWC1.txt	s17A-Output8.txt	sWC2-Base16.txt
inWC2.txt	s18A-Base16.txt	sWC2-Base32.txt
s10A-Base16.txt	s18A-Base32.txt	sWC2-Base8.txt
s10A-Base32.txt	s18A-Base8.txt	sWC2-Output16.txt
s10A-Base8.txt	s18A-Output16.txt	sWC2-Output32.txt
s10A-Output16.txt	s18A-Output32.txt	sWC2-Output8.txt

Notes:

Filename conventions	File usage notes
inXXXX.txt	Input text file. Contains text with a terminating NEWLINE (0x'0a') NEWLINE in *c* is "\n".
sXXX-BaseXX.txt	The expected output for each corresponding input file. For example, given in10A.txt as input for an 8 bit checksum, the baseline output is s10A-Base8.txt. Each of the input files has a corresponding baseline output.
sXXX-OutputXX.txt	The captured output file for the given input file and checksum size.

Please note that the following:

1. Every input file has a single line of text terminated by the hex character '0A' or the NEWLINE character.
2. Some input files are less than 80 characters long, others aren't.
3. More testing files are supplied than are used in the **hw2Test.sh** script.
4. After uploading the testing shell script (and corresponding files) remember to execute the command **chmod +x *.sh** to grant execution privileges for the script.
5. The script is executed at the command line as follows:
   ./hw2Test.sh checksum.c where the checksum program filename has the correct extension for your submission.