CET440: Computer Networking

Instructor: Dr. Weifeng Chen

Acknowledgement: thank Professor Sumey's materials

Lab assignment 3

Due on Thursday (09/22/2022) 11:59pm

Assignment type: team-based

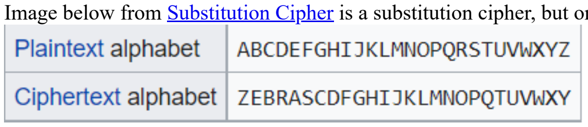
Objective: to comprehend the substitution cipher and modularization methods.

Procedure:

1. Read some basic info about Substitution Cipher

2. Write, compile and debug a C program to

   1. create a *correct Substitution cipher* for the 95 keyboard printable characters with ASCII code from 32 to 126. Note: a correct Substitution is a ***random*** *and* ***one-to-one*** mapping, i.e., any printable character is randomly mapped to a unique printable character and no two printable characters are mapped to a same printable character.
      1. Image below from Substitution Cipher is a substitution cipher, but only for the 26 upper-case letters, ASCII codes from 65 to 90.



Plaintext alphabet	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Ciphertext alphabet	ZEBRASCDFGHIJKLMNOPQTUVWXY

For example, "A" is mapped to "Z". If using their ASCII codes, 65 is mapped to 90, denoted as (65, 90). Similarly, "B" to "E" is (66, 69); "C" to "B" is (67, 66), ...

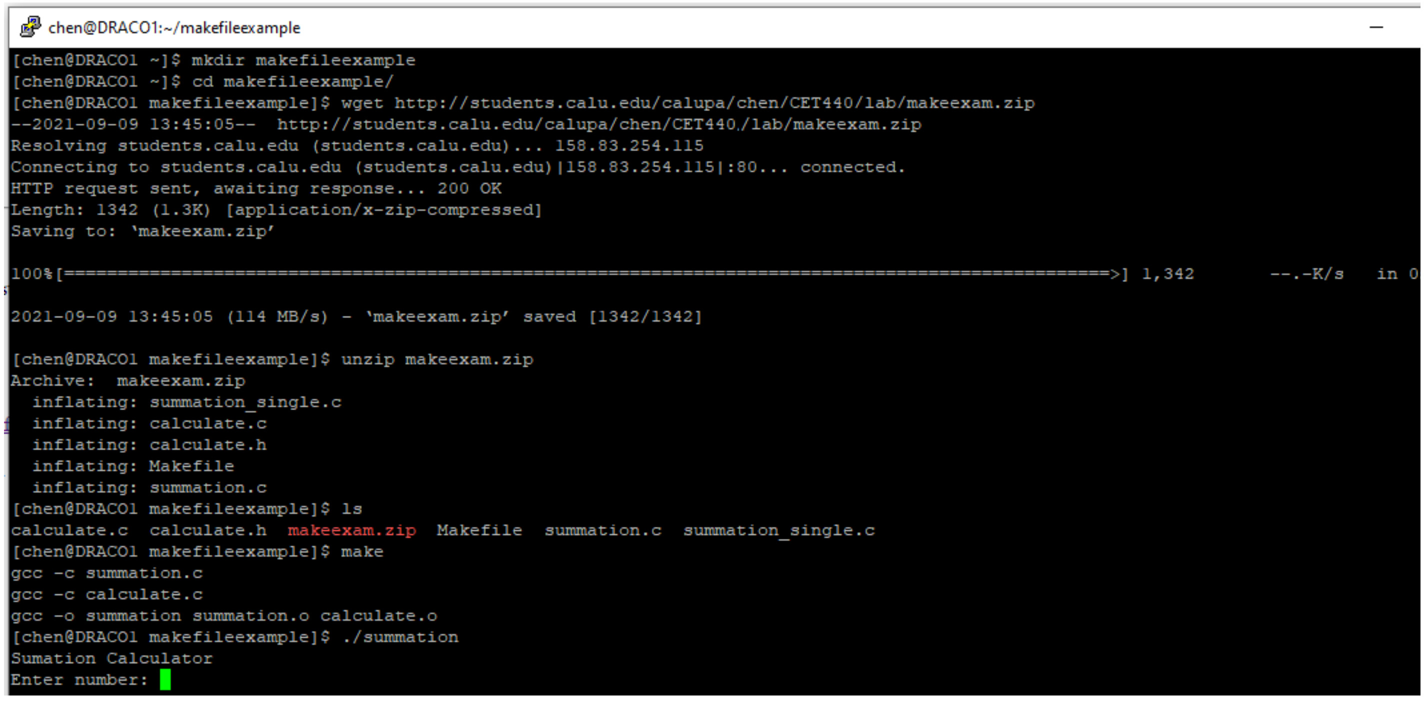
      2. **Hints:** Refer to example random.c which prints out 15 random integers from 1 to 15. Notes that the 15 random integers may have repeated values. When these 15 random integers are all different, that will be a random and one-to-one mapping. For examples, if the 15 random integers are 13, 1, 4, 2, 14, 10, 9, 8, 3, 5, 15, 6, 7, 11, 12, then the one-to-one mapping is (1, 13), (2, 1), (3, 4), (4, 2), (5, 14), (6, 10), (7, 9), (8, 8), (9, 3), (10, 5), (11, 15), (12, 6), (13, 7), (14, 11), and (15, 12).
   2. ask a user to input a string, then print out the corresponding ciphertext based on the Substitution created in step 1. For example, if a user input "WOWSOINTERESTING", using the substituion cipher above, the ciphertext will be "VLVPLFKQAOAPQFKC"
   3. print out the key, i.e., the random/one-to-one mapping like the example above

3. Study the modularization using the following example, and modularize your C program. For example, you can put all the functions that generate the subsitution and do the encryption in a module separate from the main.c.

- Example

```
mkdir makefileexample
```
//you may want to create a separate folder "makefileexample" to download and extract the zip file
```
cd makefileexample
wget http://students.calu.edu/calupa/chen/CET440/lab/makeexam.zip
unzip makeexam.zip
```
//which will extract *Makefile, summaztion_single.c, summaztion.c, calculate.h* and *calculate.c* for you.
```
make
```

```
./summation
```



chen@DRACO1:~/makefileexample

```
[chen@DRACO1 ~]$ mkdir makefileexample
[chen@DRACO1 ~]$ cd makefileexample/
[chen@DRACO1 makefileexample]$ wget http://students.calu.edu/calupa/chen/CET440/lab/makeexam.zip
--2021-09-09 13:45:05--  http://students.calu.edu/calupa/chen/CET440/lab/makeexam.zip
Resolving students.calu.edu (students.calu.edu)... 158.83.254.115
Connecting to students.calu.edu (students.calu.edu)|158.83.254.115|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1342 (1.3K) [application/x-zip-compressed]
Saving to: 'makeexam.zip'

100%[==========================================================================================>] 1,342       --.-K/s   in 0

2021-09-09 13:45:05 (114 MB/s) - 'makeexam.zip' saved [1342/1342]

[chen@DRACO1 makefileexample]$ unzip makeexam.zip
Archive:  makeexam.zip
  inflating: summation_single.c
  inflating: calculate.c
  inflating: calculate.h
  inflating: Makefile
  inflating: summation.c
[chen@DRACO1 makefileexample]$ ls
calculate.c  calculate.h  makeexam.zip  Makefile  summation.c  summation_single.c
[chen@DRACO1 makefileexample]$ make
gcc -c summation.c
gcc -c calculate.c
gcc -o summation summation.o calculate.o
[chen@DRACO1 makefileexample]$ ./summation
Sumation Calculator
Enter number:
```

Hints:

1. You can download an executable file to DRACO1 and run it to see how a correct solution would look like

```
wget https://students.calu.edu/calupa/chen/cet440/lab/ciphersub.gz
gunzip ciphersub.gz
chmod 755 ciphersub
./ciphersub
```

Deliverables: Submit the following files to D2L Dropbox "Lab Assignment 3"

1. Your .c file with comments at the beginning with your team members' names and any Acknowledgement/Credits you want to give
2. A .doc or .pdf file containing an instruction to compile & execute your program, with a screenshot showing it is running correctly, and an overview of your modules. Specifially, include your whole .h files, put the function headers in each .c file and add comment to summarize the purpose of each function. Also add a remark if a function is used in another .c file or it uses another function from another .c file. See an example here
3. **Statement of team members' contribution**

Go Back