Lab05

1. Lab Topics

This lab covers different topics, but more emphasis is placed on good use loops, ASCII and decision branching.

2. Passphrase Security System

Once upon a time in the whimsical land of Codeville, there lived a young and imaginative programmer named Alice. Alice had a knack for creating extraordinary things with her code, and today was no different. She had been tasked with building the most secure passphrase system the kingdom had ever seen. As she sat in her cozy room, surrounded by books and her trusty laptop, she began to ponder what would make the perfect passphrase. "Passphrases should not only be secure but also memorable," thought Alice. That's when an idea popped into her head - what if passphrases were like magical incantations?

With a mischievous grin, Alice decided to consult you about the idea since you are well known for your expertise in C++ programs. You like the idea which made you to start working on it together with Alice. She designed the algorithm in form of a pseudocode and gave you, the expert to program it such that the system can turn ordinary phrases into extraordinary passwords. The user would enter a simple phrase, and program would transform it into a complex and secure key. The explanation of Alice idea is given below.

Write a C++ program that takes 5 characters from the user. The program should iterate over each character of the sequence and perform operations based on the following criteria:

1. Each character is to be converted to ASCII values and then the instructions below should be achieved.
2. If the ASCII value is even, add 40. If the resulting value is at least 2 digits long and it falls between 32 and 126 inclusive, convert the integer to ASCII character and print the result. Else, print the result of the addition.
3. If the ASCII value is odd, subtract 70. If subtracting 70 results in a negative integer, square the result, subtract 40, if the resulting integer is at least 2 digit long and it falls between the range of 32 to 126 inclusive, convert the integer to ASCII character and print the result. Else, print the result of the subtraction.

Requirements:

1. Use a do-while loop to repeatedly prompt the user for a five-character sequence until a valid input is provided in terms of length.
2. Use a for loop to iterate over each digit of the input number.

3. Implement decision structures (if-else statements) to determine the operation for each digit based on the criteria mentioned.
4. Use the break statement to exit the loop when the operation is complete for all digits.
5. Use the continue statement to skip specific iterations when the digit is not processed (e.g., if the digit is not odd or even).
6. Use an infinite loop to continuously run the program until the user chooses to exit.
    a. Use a do while loop.
    b. Ask if the user want to continue running the program. If the user's input is Yes, YES, yes, Y, **or** y, continue running the program, else end the program "gracefully".

General Hints

1. You can use a simple concept of checking if the value is between 100 and 999 inclusive to checking for 3 digits long and use the same idea for 2 digits long (10 to 99 inclusive).
2. Use appropriate arithmetic operators, relational and logical operators to determine the results for the values.
3. For this program, we are assuming that the user will can enter **any character** as input.
4. **DO NOT** use arrays, structs, classes or any other concept not yet thought in class.

BONUS (5 points):
Write a reusable function for part 6 of the requirement.
Note: No help will be provided for this bonus. Students are expected to research about it and learn how to do it.

Example Output 1:
```
Enter a five-character sequence: wer./
w => 49
e => 31
r => 154
. => V
/ => 489
Do you want to continue? (yes/no): n
```

Example Output 2 (Testing the program):

```
Enter a five-character sequence: ,.;'/]d
Invalid input. Please enter exactly five characters.
Enter a five-character sequence: ,./;'
, => T
. => V
/ => 489
; => Q
' => 921
Do you want to continue? (yes/no): Y
Enter a five-character sequence: }\\ ;
```

```
Invalid input. Please enter exactly five characters.
Enter a five-character sequence: Invalid input. Please enter
exactly five characters.
Enter a five-character sequence:   ./'
Invalid input. Please enter exactly five characters.
Enter a five-character sequence: ./ .'
Invalid input. Please enter exactly five characters.
Enter a five-character sequence: Invalid input. Please enter
exactly five characters.
Enter a five-character sequence: <.?;'
< => d
. => V
? => 9
; => Q
' => 921
Do you want to continue? (yes/no): n
```

Example Output 3:

```
Enter a five-character sequence: #$%^(
# => 1185
$ => L
% => 1049
^ => 134
( => P
Do you want to continue? (yes/no): Y
Enter a five-character sequence: _+)lK
_ => 25
+ => 689
) => 801
l => 148
K => 5
Do you want to continue? (yes/no): y
Enter a five-character sequence: ?Ja<.
? => 9
J => r
a => 27
< => d
. => V
Do you want to continue? (yes/no): n
```

3. How to get full marks

To get a 100% on this lab your code should:

1. Use good variable names such that one can easily understand a variable's purpose just by looking at the name.
2. The program needs to be intuitive (e.g., display proper messages while you are taking user input or printing the result)
3. Follow all good coding conventions such as proper indentation.

4. Adhere to all coding standards outlined in lab2.
5. Follow the instructions of cloning, making dir, and submitting your code to git as previously discussed in lab01 and lab02
6. Comment your code properly (do not write comments for things that are obvious)
7. Push your most recent code in git and submit through canvas as well. The canvas submission should include following files:
   a. **The cpp file downloaded from git**
   b. **At least 2 Image files**
      i. **Screenshot of the right result**
      ii. **Screenshot of testing "abnormal" values for the user's input.**