CISS240: Introduction to Programming
Assignment 3

Name: _____

Objectives

- Declare integer variables
- Use integer operators
- Input and output of integer values

Q1. It's a well–known fact that when you take a positive integer and compute its remainder after dividing it by 9, you will find that it's the same as first adding the digits of the number and then computing the remainder after dividing by 9. For instance the remainder of 38 after dividing by 9 is 2 (because 38 is $4 \times 9 + 2$) and the sum of the digits of 38 is $3 + 8 = 11$ whose remainder after dividing by 9 is also 2 (because $11 = 1 \times 9 + 2$). Obviously it's easier to compute the remainder of $3 + 8$ when divided by 9 than the remainder of 38 when divided by 9. This fact is used by some to do mental calculations quickly. The goal of this question is to verify the above fact for 4–digit numbers.

Write a program that verifies the above fact for 4–digit numbers. Your program must pass the following tests. The first line (underlined) is the input from the user. The second line has three output values:

- The first is the remainder of the user–input integer after dividing by 9
- The second is the sum of the digits of user–input integer, and
- The third is the remainder of the second number after dividing it by 9.

If the above fact is true, the first output value must be the same as the third output value. For instance in Test 1, the first output value is 1 and the third is 1.

(With more math one can prove that the above fact is in fact true for a positive integer of any number of digits. For this question we only want to verify it for the cases specified by the user.)

Test 1.

```
1234
1 10 1
```

Test 2.

```
2468
2 20 2
```

Test 3.

```
1111
4 4 4
```

Q2. Write a program that prompts the user for 5 digits, computes the integer value with the given digits, stores this integer value in a variable x, and prints the value of this variable. You must use this skeleton code for `main()`:

```
// Prompt user for 5 digits and store in d4,d3,d2,d1,d0
int d4 = 0, d3 = 0, d2 = 0, d1 = 0, d0 = 0;


// Compute and store the integer value with digits d4,d3,d2,d1,d0 in x
int x = ____________________________________;


// Print x

return 0;
```

Test 1.

```
1 2 3 4 5
12345
```

Test 2.

```
4 2 0 4 1
42041
```

Test 3.

```
0 0 0 4 2
42
```

Test 4.

```
0 0 0 0 1
1
```

Q3. Write a program that prompts the user for a 5–digit integer, encrypts it, and finally displays the original integer (known as the plaintext) and the encrypted integer (known as the ciphertext). The following is the algorithm for our encryption.

Suppose the digits are a, b, c, d, and e where e is the rightmost digit (also known as the least significant digit). Let the digits of the encrypted integer be $a1$, $b1$, $c1$, $d1$, and $e1$. These digits make up the encrypted integer where $e1$ is the rightmost digit.

- First: Add 1 to e to get $e1$. If $e1$ becomes 10, change $e1$ to 0.
- Second: Swap d and c into $c1$ and $d1$. In other words the value of $c1$ is the value of d and the value of $d1$ is the value of c.
- Third: Swap a and b into $b1$ and $a1$. In other words the value of $b1$ is the value of a and the value of $a1$ is the value of b.

Refer to the test cases for examples.

[HINT: You want to perform this experiment before trying this program. Use C++ to compute (and display) the $0 \,\%\, 10$, $1 \,\%\, 10$, $2 \,\%\, 10$, $3 \,\%\, 10$, $\ldots$, $10 \,\%\, 10$. Why is this useful for the first step above?]

You must use the following skeleton code for `main()`:

```
// Prompt user for an integer value for variable plaintext.
int plaintext = 0;

// a, b, c, d, e are the digits of plaintext

// a1, b1, c1, d1, e1 are the digits of the encrypted ciphertext

// Form the ciphertext (an int)

// Print plaintext and ciphertext

return 0;
```

Test 1.

```
12345
12345 21436
```

Test 2.

```
97531
97531 79352
```

Test 3.

```
56789
56789 65870
```

Q4. In this program you will prompt the user for two times during the day and print the difference. The user will input a time as a 6–digit integer value where the first 2 digits represent the hour, the next 2 represent the minute, and the last 2 represent the seconds. For instance, the integer value 091503 represents the time 09:15:03 in the 24–hour format, i.e, 3 seconds after 9:15 AM; the integer value 131503 represents the time 13:15:03 in the 24–hour format, i.e, 3 seconds after 1:15 PM. While the user input is as a 6–digit number, the output of the difference must be written in the 24–hour format as shown above. For instance the 24–hour format of 3 seconds after 9:15 AM is 09:15:03; the 24–hour format of 3 seconds after 1:15 PM is 13:15:03.

Test 1.

```
091400
091400
00:00:00
```

Test 2.

```
091400
091415
00:00:15
```

Test 3.

```
091400
091500
00:01:00
```

Test 4.

```
091410
091525
00:01:15
```

Test 5.

```
091430
091500
00:00:30
```

Test 6.

```
091400
092500
00:11:00
```

Test 7.

```
091410
092500
00:10:50
```

Test 8.

```
100000
110000
01:00:00
```

Test 9.

```
100000
110102
01:01:02
```

Test 10.

```
100000
110100
01:01:00
```

Test 11.

```
100000
110002
01:00:02
```

Test 12.

```
100201
110502
01:03:01
```

Test 13.

```
101010
111202
01:01:52
```

Test 14.

```
101010
110502
00:54:52
```