Lab 2: Complex Calculations, Decision Making and Debugging a C Program

Eric Cao – 1319418

Partner: Jeremiah George

# Part 1

## Problem – Objective

Create a program that prompts the user for an integer greater than 0 and smaller than 15, which is then converted to binary form digit by digit. The result is then outputted to the terminal line by line.

## Assumptions and Constraints

* The input from the user is an integer.
* The number received from the user is between 0 and 15.

## Pseudocode

Output -> statement prompting for a number between 0 and 15

(Integer) decimal <- received input

If 0 < decimal < 15 then stop program

Last digit = remainder(decimal/2)

Decimal = decimal / 2

Third digit = remainder(decimal/2)

Decimal = decimal / 2

Second digit = remainder(decimal/2)

Decimal = decimal / 2

First digit = decimal

Output ->

“The four digits of” + decimal + “are as follows: (newline)

Most significant digit: ” + first digit + “(newline)

Next digit: ” + second digit + “(newline)

Next digit: ” + third digit + “(newline)

Least significant digit: ” + last digit

# Part 2

## Problem – Objective

Create a program that receives as input, the x-y coordinates of two vectors A and B. The angle formed at the origin between A and B is then calculated in both radians and degrees using the cosine law, which is then displayed to the terminal.

## Assumptions and Constraints

* The received input consists of four real numbers for the x and y coordinates of point A, then the x and y coordinates of point B.
* The angle formed between A and B is at the origin.
* The output consists of the angle in radians with three decimal places, then in degrees with two decimal places.

## Pseudocode

Output -> prompt for x and y values of point A in the format (x,y)

Coordinates x\_A, y\_A <- received input

Output -> prompt for x and y values of point B in the format (x,y)

Coordinates x\_B, y\_B <- received input

// calculate dot product:

Dot product = x\_A \* x\_b + y\_A, y\_B

// calculate magnitude of vectors:

Magnitude A = sqrt(x\_A^2 + y\_A^2)

Magnitude B = sqrt(x\_B^2 + y\_B^2)

// divide dot product by both magnitudes of vectors

cosine of angle = (dot product) / (magnitude A \* magnitude B)

angle radians = arccos(cosine of angle)

angle degrees = (angle radians) \* (180/pi)

Output -> (angle radians, three decimal places) and (angle degrees, two decimal places)

# Part 3

## Problem – Objective

Create a program that receives a number representing the total minutes to be converted into hours. The remaining minutes are rounded to the nearest multiple of 15, or quarter hour. The output to terminal is the rounded minutes and hours.

## Assumptions and Constraints

* The input received representing the total minutes is a positive integer.
* The output consists of a positive integer for the number of hours and either 0, 15, 30, or 45 minutes for the number of minutes rounded to the quarter hour.

## Pseudocode

Output -> prompt for total number of minutes

Total minutes <- received input

Hours = (total minutes) / 60

Minutes = remainder(total minutes/60)

If (minutes <= 7) then rounded minutes = 0

Else if (minutes <= 22) then rounded minutes = 15

Else if (minutes <= 37) then rounded minutes = 30

Else if (minutes <= 52) then rounded minutes = 45

Else {

Rounded minutes = 0

Hours = hours + 1

}

Output -> hours, then rounded minutes

# Part 4

## Problem – Objective

Create a program that receives a number from the terminal, and outputs the sum of every digit in the number.

## Assumptions and Constraints

* The input received is an integer from -99999 to 99999, inclusive.
* The program exits after outputting the sum of the digits or if the number is not within the range.

## Pseudocode

Output -> prompt for number between -99999 to 99999

Number <- received input

Number = absolute value(number)

If (number > 99999) then {

Output -> statement stating number is out of range

Exit program

}

Ten-thousands digit = number / 10000

Number = remainder(number / 10000)

Thousands digit = number / 1000

Number = remainder(number / 1000)

Hundreds digit = number / 100

Number = remainder(number / 100)

Tens digit = number / 10

Number = remainder(number / 10)

Ones digit = number

Output -> “The sum of digits is: ” + ten-thousands digit + “ + ” + thousands digit + “ + ” + hundreds digit + “ + ” + tens digit + “ + ” + ones digit + “ = ” + (ten-thousands digit + thousands digit + hundreds digit + tens digit + ones digit)

# Part 5

## Problem – Objective

Debug a given C file to achieve a desired result, where the program receives a 4-digit combination from the user and converts it into a “decrypted” combination as described in the file’s comments.

## Assumptions and Constraints

* The combination is a 4-digit positive integer.
* Once the decrypted combination is output to the terminal, the program exits.

## Errors and Modifications

* Line 1: the phrase (< stdio .h >) is incorrectly written with three extra spaces which were then removed to form (<stdio.h>).
* Line 2: the argument (“ % d ”) passed to the scanf function is written poorly with extra spaces and is changed to (“%d”).
* Line 14, 17, 20: the variables (d4), (d3), and (d2) are being improperly assigned the remainder of the unprocessed 4-digit combination instead of being assigned to the encrypted combination (encComb). The variables (d4), (d3), and (d2) are replaced with (encComb).
* Line 12, 15, 18: the variable (encComb) is being improperly assigned the digit extracted from the 4-digit combination when it should instead be assigned to their respective digits. The assignment on line 12 is changed to (d4), on line 15 is changed to (d3), and on line 18 is changed to (d2).
* Line 25: a. the phrase (\ n) is improperly written and is replaced with (\n); b. the phrase (% d) is improperly written and replaced with (%d).

# Conclusion

In this lab, I learned to use several functions included in the math.h header like sqrt and pow. I also learned how to use the modulo operator to remove digits from a larger integer, as well as using a if/elseif/else statement to create branches depending on the value of certain variables. Going forward, I want to be more familiar with these functions and operators in order to write code more easily and at a faster pace.