



# Digital Design Verification

## Lab # 02

## Decision Making, Conditionals

**Release: 1.1**

**Date: 30-July-2024**

**NUST Chip Design Centre (NCDC), Islamabad, Pakistan**

## Revision History

| Revision Number | Revision Date | Nature of Revision | By |
|---|---|---|---|
| 1.0 | 2/05/2024 | Complete manual | Dr. Abid |
| 1.1 | 30/07/2024 | Revision in manual | Hira Sohail |

# Contents

[Azad's Law: The one facilitating cheating will be punished with 0 marks]

## OBJECTIVE:

The objective of this lab is to enable students to answer following questions:

- How to use flow charts to devise algorithms for solving problems?
- What sort of problems require decision making?
- How to program computers to solve such problems?
- How does C/C++ support decision make?

## TOOLS:

- GCC
- GDB

## TASK # 01:

Find out whether a particular year is a leap year or not? Consider some authentic source for figuring out leap year such as NASA.

***Flow Chart*** **:** 10 minutes ***Coding*** **:** 10 minutes

## TASK # 02:

Take latitude and longitude in degree minute and seconds and convert it into decimal form. Figure out whether it is above the equator or not?

***Flow Chart*** **:** 15 minutes ***Coding*** **:** 15 minutes

## TASK # 03:

Take a 3-digit number from the user and print out the digit at unit, 10th and 100th place.

***Flow Chart*** **:** 5 minutes ***Coding*** **:** 10 minutes

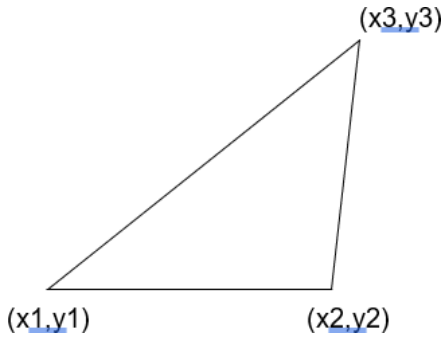
## TASK # 04:

Take any five numbers from the user and find out mean, median, mode and standard deviation?

**Flow Chart :** 5 minute **Coding :** 10 minute

## TASK # 05:

Take three points from the user (x1,y1), (x2,y2) and (x3, y3). Figure out whether these three points constitute a right-angle triangle?




***Flow Chart :*** 10 *minute* ***Coding :*** 10 minute

## TASK # 06:

Imagine you are playing a grid (4x4) game where the player can move in four directions Up, Down, Right and Left. Take the initial position of the player from the user e.g. (1,2) and position of the obstacle e.g. (1,3). Now take the direction in which the player will move. Figure out whether it will collide with an obstacle or not? You need to check the boundary conditions.

| | | | |
|---|---|---|---|
| (1,1) | | | (4,1) |
| | | | |
| | | | |
| (1,4) | | | (4,4) |

***Flow Chart:*** 15 minutes ***Coding:*** 25 minutes

## TASK # 07:
## Bit Operations

For this task, you will complete bit_ops.c, available in the attached zip files folder. By implementing the bit manipulation functions get_bit, set_bit, and flip_bit (shown below). You may ONLY use bitwise operations such as and (&), or (|), xor (^), not (~), left shifts (<>). You may not use any for/while loops or conditional statements. You also

may not use modulo (%), division, addition, subtraction, or multiplication for this question.

Finish implementing get_bit, set_bit, and flip_bit.

Once you complete these functions, you can compile and run your code using the following commands:

```
Gcc bit_ops.c test_bit_ops.c -o bit_ops
```

```
./bit_ops
```

This will print out the results of the tests. Make sure that you do not use any of the forbidden operations before running the code.

## SUBMISSION:

Please submit all the flow charts in hard copy and .c files of all the tasks along with the screenshots of outputs on LMS in a proper report.