Digital Design Verification

FYP/Internship Program

Review of C constructs

Lab # 02

Release: 1.1

Date: 30-July-2024

NUST Chip Design Centre (NCDC), Islamabad, Pakistan

Revision History

Revision Number	Revision Date	Nature of Revision	Approved 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

NOTE:
First make a flowchart on paper for the task and then go for coding. Add clear scanned images of all the flow charts on your report.

TASK # 01:

Treasure Hunt: Imagine there is a ground which is represented by a 2D grid as shown below. There is a treasure hidden in one of the locations. You have to figure out where exactly the treasure is.

- Your program will not take any input from the user.
- It will generate the location of the hidden treasure (xt,yt) using a random number. You have used random numbers in the previous lab. Choose appropriate values to generate random location within the grid.
- Now search the grid using loops to figure out what is the location of the hidden treasure. Once you found it, print "Hurrah!, I have found the hidden treasure".

(1,1)			(4,1)
(1,4)			(4,4)

TASK # 02:

Stopwatch Design: Design a stop watch which counts the number of minutes (N). Ask the user to enter the number of minutes.

Note: The watch should display the counter like this. If N = 2, it should count up to 2 minutes.

M : S (M = Minutes, S= Seconds)

00 : 00

00 : 01

00 : 02

00 : 03

|

|

|

00 : 59

01 : 00

01 : 01

|

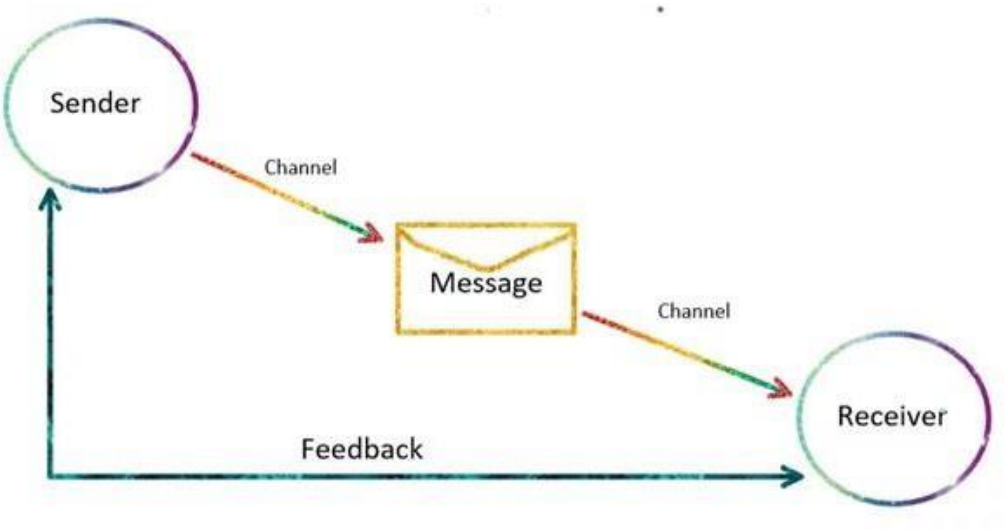
|

|

01 : 59

TASK # 03:

Imagine a communication system where a sender sends some message in the form of a data packet to a remote receiver. In order to ensure that the correct packet reaches a remote destination, the sender generates a checksum of the data and appends it with the packet. Upon reception, the receiver also generates the checksum using the same algorithm and matches with the checksum sent by the sender. If both match, it means data is correct.



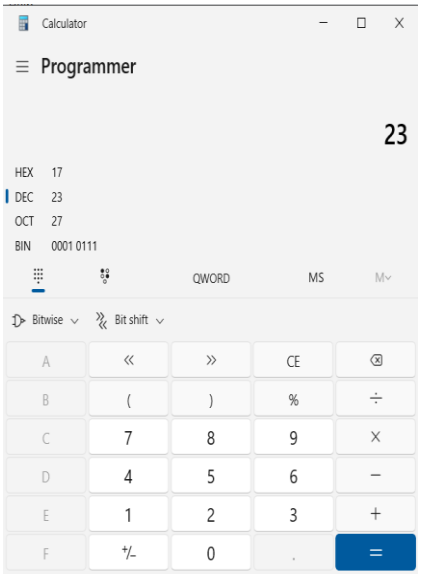
- Take the number of bytes as input from the user
- Take the byte one by one from the user
- Generate the checksum by computing the **XOR** of all the bytes. You can take XOR of a and b using **a ^ b**.

TASK # 04:

Number Converter

Take a number N in decimal format from the user and convert it into

- Hexadecimal format
- Binary Format
- Octal Format



TASK # 05:

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

TASK # 06:

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:

Add scanned images of all the flow charts along with the screenshots of outputs on LMS in a proper report. Submit .c files of all the tasks along with the report.