



# Digital Design Verification

## FYP/Internship Program

### Review of C constructs

**Lab # 02**

**Name: Abdul Hadi Afzal**

**Date: 25-02-2025**

# Table of Contents

## TASK 1:

**CODE:**

```
/*************************************************
 * Author: AbdulHadi Afzal
 * Task: Treasure Hunt Simulation
 *************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    int xt, yt; // Treasure coordinates

    // Seed the random number generator
    srand(time(0));

    // Generate random coordinates for the treasure (between 1
and 4)
    xt = (rand() % 4) + 1;
    yt = (rand() % 4) + 1;

    printf("Searching for treasure...\n");

    // Searching the grid
    for (int x = 1; x <= 4; x++) {
        for (int y = 1; y <= 4; y++) {
            printf("Checking location (%d, %d)...\n", x, y);
            if (x == xt && y == yt) {
                printf("Hurrah! I have found the hidden treasure
at (%d, %d)!\n", xt, yt);
                return 0;
            }
        }
    }

    return 0;
}
```

**SS:**

```
# ./task1.exe
Searching for treasure...
Checking location (1, 1)...
Checking location (1, 2)...
Checking location (1, 3)...
Checking location (1, 4)...
Hurrah! I have found the hidden treasure at (1, 4)!
```

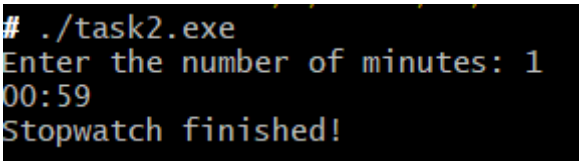
## TASK 2:

**CODE:**

```
/*************************************************
```

```
 * Author: AbdulHadi Afzal
 * Task: Stopwatch Simulation
 ************************************************/

#include <stdio.h>
#include <unistd.h>
#include <windows.h>

int main() {
    int minutes;

    printf("Enter the number of minutes: ");

    if (scanf("%d", &minutes) != 1) {
        printf("Invalid input! Please enter an integer
value.\n");
        return 1;
    }

    // Simulating the stopwatch
    for (int m = 0; m < minutes; m++) {
        for (int s = 0; s < 60; s++) {
            printf("\r%02d:%02d", m, s);
            fflush(stdout);
            Sleep(1000);
        }
    }

    printf("\nStopwatch finished!\n");
    return 0;
}
```

**SS:**




## TASK 3:

**CODE:**

```
/************************************************
 * Author: AbdulHadi Afzal
 * Task: Checksum Validation for Data Integrity
 ************************************************/

#include <stdio.h>

int main() {
    int n, i, checksum = 0, received_checksum,
calculated_checksum = 0;

    printf("Enter the number of bytes in the data packet: ");
```

```
    scanf("%d", &n);

    int data[n];

    printf("Enter %d bytes (as integers):\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &data[i]);
        checksum ^= data[i];
    }

    printf("Generated Checksum: %d\n", checksum);

    printf("\nReceiver Side:\n");
    printf("Enter the received %d bytes (including checksum):\n",
n + 1);

    for (i = 0; i <= n; i++) {
        scanf("%d", &received_checksum);
        calculated_checksum ^= received_checksum;
    }

    if (calculated_checksum == 0) {
        printf("Data is CORRECT (Checksum Matched)\n");
    } else {
        printf("Data is CORRUPTED (Checksum Mismatch)\n");
    }

    return 0;
}
```

**SS:**

```
Enter the number of bytes in the data packet: 8
Enter 8 bytes (as integers):
1
2
3
4
5
6
7
8
Generated Checksum: 8

Receiver Side:
Enter the received 9 bytes (including checksum):
1
2
3
4
5
6
7
8
8
Γ£à Data is CORRECT (Checksum Matched)
```

## TASK 4:

**CODE:**

```
/*************************************************
 * Author: AbdulHadi Afzal
 * Task: Number System Converter (Decimal to Binary, Octal, Hex)
 *************************************************/

#include <stdio.h>

// Function to convert and print binary representation
void printBinary(int n) {
    if (n > 1)
        printBinary(n / 2);
    printf("%d", n % 2);
}

int main() {
    int number;

    printf("Enter a decimal number: ");
    scanf("%d", &number);

    printf("Hexadecimal: %X\n", number);
    printf("Octal: %o\n", number);

    printf("Binary: ");
    printBinary(number);
    printf("\n");

    return 0;
}
```

**SS:**

```
# ./task4.exe
Enter a decimal number: 10
Hexadecimal: A
Octal: 12
Binary: 1010
```

## TASK 5:

**CODE:**

```
/*********************************************************
 * Author: AbdulHadi Afzal
 * Task: Statistical Calculator (Mean, Median, Mode, Std Dev)
 *********************************************************/

#include <stdio.h>
#include <math.h>

#define SIZE 5
```

```
// Function to calculate mean
double mean(int arr[]) {
    double sum = 0;
    for (int i = 0; i < SIZE; i++) {
        sum += arr[i];
    }
    return sum / SIZE;
}

// Function to calculate median
double median(int arr[]) {
    for (int i = 0; i < SIZE - 1; i++) {
        for (int j = 0; j < SIZE - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
    return arr[SIZE / 2];
}

// Function to calculate mode
int mode(int arr[]) {
    int maxCount = 0, modeVal = arr[0];
    for (int i = 0; i < SIZE; i++) {
        int count = 0;
        for (int j = 0; j < SIZE; j++) {
            if (arr[j] == arr[i]) count++;
        }
        if (count > maxCount) {
            maxCount = count;
            modeVal = arr[i];
        }
    }
    return (maxCount == 1) ? -1 : modeVal;
}

// Function to calculate standard deviation
double standardDeviation(int arr[]) {
    double m = mean(arr), sum = 0;
    for (int i = 0; i < SIZE; i++) {
        sum += pow(arr[i] - m, 2);
    }
    return sqrt(sum / SIZE);
}

int main() {
    int numbers[SIZE];

    printf("Enter 5 numbers: ");
    for (int i = 0; i < SIZE; i++) {
        scanf("%d", &numbers[i]);
```

```
    }

    printf("Mean: %.2f\n", mean(numbers));
    printf("Median: %.2f\n", median(numbers));

    int modeVal = mode(numbers);
    if (modeVal == -1)
        printf("Mode: No mode (all numbers appear once)\n");
    else
        printf("Mode: %d\n", modeVal);

    printf("Standard Deviation: %.2f\n",
standardDeviation(numbers));

    return 0;
}
```

SS:

```
Enter 5 numbers: -923
-34
5
0
5
Mean: -189.40
Median: 0.00
Mode: 5
Standard Deviation: 367.09
```

## TASK 6:

**CODE:**

```
/*************************************************************
* Author: AbdulHadi Afzal
* Task: Bitwise Operations (Get, Set, Flip Bits)
*************************************************************/

#ifndef BIT_OPS_H
#define BIT_OPS_H

unsigned get_bit(unsigned x, unsigned n);
void set_bit(unsigned *x, unsigned n, unsigned v);
void flip_bit(unsigned *x, unsigned n);

#endif

#include "bit_ops.h"

// Returns the Nth bit of X
unsigned get_bit(unsigned x, unsigned n) {
return (x >> n) & 1;
}

// Sets the Nth bit of X to V
```

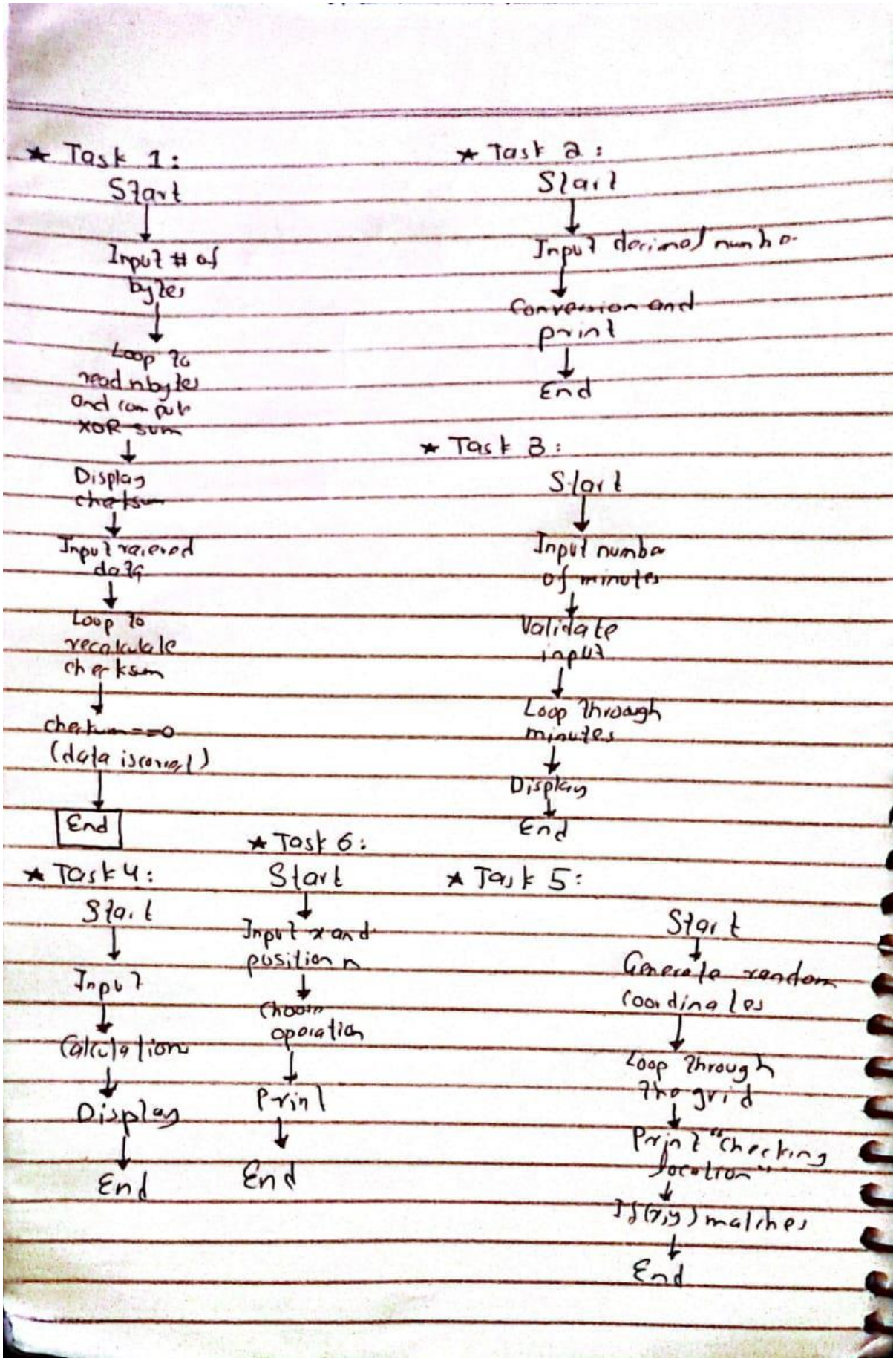
```
void set_bit(unsigned *x, unsigned n, unsigned v) {
if (v == 1)
*x |= (1 << n);
else
*x &= ~(1 << n);
}

// Flips the Nth bit in X
void flip_bit(unsigned *x, unsigned n) {
*x ^= (1 << n);
}

#include <stdio.h>
#include "bit_ops.h"

void test_get_bit(unsigned x, unsigned n, unsigned expected) {
unsigned a = get_bit(x, n);
if (a != expected) {
printf("get_bit(0x%08x,%u) returned 0x%08x, but we expected
0x%08x\n", x, n, a, expected);
} else {
printf("get_bit(0x%08x,%u) returned 0x%08x, correct\n", x, n, a);
}
}

void test_set_bit(unsigned x, unsigned n, unsigned v, unsigned
expected) {
unsigned o = x;
set_bit(&x, n, v);
if (x != expected) {
printf("set_bit(0x%08x,%u,%u) returned 0x%08x but we expected
0x%08x\n", o, n, v, x, expected);
} else {
printf("set_bit(0x%08x,%u,%u) returned 0x%08x, correct\n", o, n,
v, x);
}
}

void test_flip_bit(unsigned x, unsigned n, unsigned expected) {
unsigned o = x;
flip_bit(&x, n);
if (x != expected) {
printf("flip_bit(0x%08x,%u) returned 0x%08x, but we expected
0x%08x\n", o, n, x, expected);
} else {
printf("flip_bit(0x%08x,%u) returned 0x%08x, correct\n", o, n,
x);
}
}

int main(int argc, const char *argv[]) {
printf("\nTesting get_bit()\n\n");
test_get_bit(0b1001110, 0, 0);
test_get_bit(0b1001110, 1, 1);
```

```
test_get_bit(0b1001110, 5, 0);
test_get_bit(0b11011, 3, 1);
test_get_bit(0b11011, 2, 0);
test_get_bit(0b11011, 9, 0);

printf("\nTesting set_bit()\n\n");
test_set_bit(0b1001110, 2, 0, 0b1001010);
test_set_bit(0b1101101, 0, 0, 0b1101100);
test_set_bit(0b1001110, 2, 1, 0b1001110);
test_set_bit(0b1101101, 0, 1, 0b1101101);
test_set_bit(0b1001110, 9, 0, 0b1001110);
test_set_bit(0b1101101, 4, 0, 0b1101101);
test_set_bit(0b1001110, 9, 1, 0b1001001110);
test_set_bit(0b1101101, 7, 1, 0b11101101);

printf("\nTesting flip_bit()\n\n");
test_flip_bit(0b1001110, 0, 0b1001111);
test_flip_bit(0b1001110, 1, 0b1001100);
test_flip_bit(0b1001110, 2, 0b1001010);
test_flip_bit(0b1001110, 5, 0b1101110);
test_flip_bit(0b1001110, 9, 0b1001001110);

printf("\n");
return 0;}
```

**SS:**

```
Testing get_bit()

get_bit(0x0000004e,0)returned 0x00000000, correct
get_bit(0x0000004e,1)returned 0x00000001, correct
get_bit(0x0000004e,5)returned 0x00000000, correct
get_bit(0x0000001b,3)returned 0x00000001, correct
get_bit(0x0000001b,2)returned 0x00000000, correct
get_bit(0x0000001b,9)returned 0x00000000, correct

Testing set_bit()

set_bit(0x0000004e,2,0) returned 0x0000004a, correct
set_bit(0x0000006d,0,0) returned 0x0000006c, correct
set_bit(0x0000004e,2,1) returned 0x0000004e, correct
set_bit(0x0000006d,0,1) returned 0x0000006d, correct
set_bit(0x0000004e,9,0) returned 0x0000004e, correct
set_bit(0x0000006d,4,0) returned 0x0000006d, correct
set_bit(0x0000004e,9,1) returned 0x0000024e, correct
set_bit(0x0000006d,7,1) returned 0x000000ed, correct

Testing flip_bit()

flip_bit(0x0000004e,0) returned 0x0000004f, correct
flip_bit(0x0000004e,1) returned 0x0000004c, correct
flip_bit(0x0000004e,2) returned 0x0000004a, correct
flip_bit(0x0000004e,5) returned 0x0000006e, correct
flip_bit(0x0000004e,9) returned 0x0000024e, correct
```

## Flow Chart:


★ Task 1:
Start
Input # of bytes
Loop to read n bytes and compute XOR sum
Display checksum
Input received data
Loop to recalculate checksum
checksum==0 (data is correct)
End
★ Task 2:
Start
Input decimal number
Conversion and print
End
★ Task 3:
Start
Input number of minutes
Validate input
Loop through minutes
Display
End
★ Task 4:
Start
Input
Calculations
Display
End
★ Task 6:
Start
Input x and position n
Choose operation
Print
End
★ Task 5:
Start
Generate random coordinates
Loop through the grid
Print "Checking location"
If (x,y) matches
End

## Conclusion:

In **Lab 2**, we explored various fundamental concepts in **C programming**, focusing on bitwise operations, numerical conversions, checksum validation, and interactive simulations. Each task provided hands-on experience with important programming techniques, improving our understanding of data manipulation and control structures.