CS111 – Introduction to Computer Science (*the* Mainstream *and* Software Engineering *Programmes*)
COM101 – Introduction to Computers (*the* Data Science, Robotics Software Engineering, *and* Multimedia Software Engineering *Programmes*)

Exercises – Problems Sheet # 1: An Introduction to Computer Science & Computational Thinking

No. Of Questions: 7 **No. Of Pages: 3**

- **To be submitted during the Labs of week 5.**
- **Students will lose 2 marks if this homework is not delivered on time or found to be copied.**
- **The submitted solutions should be handwritten and NOT typed/printed.**

Question 1: Briefly describe – in your own words – the following terms:

1. A computer.
2. The four basic operations of a computer.
3. A program.
4. Computer Science.
5. Information Processing.
6. Informatics.
7. Boolean Logic.
8. An Operating System (OS).
9. The RGB System.
10. Peripheral Devices.
11. An Algorithm.
12. A Software Library

Question 2: Briefly differentiate between the following terms:

1. Data versus Information.
2. The ASCII code versus the Unicode.
3. The ROM versus the RAM.
4. Primary Storage versus Secondary Storage.
5. Natural Languages versus Programming Languages.
6. The Decimal Systems versus the Binary System.
7. Psycho-informatics versus Business Informatics (BI).
8. System Software versus Application Software.
9. Digital Image Processing versus Computer Graphics versus Computer Vision.
10. A Kilo in the Metric System versus a Kilo in the Binary System.
11. A Bit versus a Byte.
12. Syntax versus Semantics (in programming languages).
13. The Arabic Numerals versus the Roman Numerals.
14. Pseudocode versus Flowcharts.
15. Conditions/Decisions versus Loops/Repetitions (in an algorithm).
16. Critical Systems versus Real-Time Systems.

Question 3: Draw a suitable block diagram of the basic organisation of a computer hardware system and describe them in brief.

<u>Question 4:</u> Given the following solved example for calculating the value of the decimal number $(9804)_{10}$:

9		**8**		**0**		**4**		
×		×		×		×		
1000 (i.e., 10^3)		**100** (i.e., 10^2)		**10** (i.e., 10^1)		**1** (i.e., 10^0)		
=		=		=		=		
9000	+	**800**	+	**0**	+	**4**	=	**9804**

What is the value of the following binary number $(11010)_2$?

1		**1**		**0**		**1**		**0**		
×		×		×		×		×		
____ (i.e., 2^4)		____ (i.e., 2^3)		____ (i.e., 2^2)		____ (i.e., 2^1)		____ (i.e., 2^0)		
=		=		=		=		=		
____	+	____	+	____	+	____	+	____	=	____

<u>Question 5:</u> Given the following examples of counting using the Arabic Numerals:

08	**28**	**098**	**0998**
09	**29**	**099**	**0999**
10	**30**	**100**	**1000**
11	**31**	**101**	**1001**

Complete the missing binary numbers in the following table:

0 1 0 [i.e., 2]	_ _ _ _ [i.e., 6]	_ _ _ _ _ _ _ _ [i.e., 125]	_ _ _ _ _ _ _ _ [i.e., 253]
0 1 1 [i.e., 3]	**0 1 1 1** [i.e., 7]	_ _ _ _ _ _ _ _ [i.e., 126]	_ _ _ _ _ _ _ _ [i.e., 254]
_ _ _ [i.e., 4]	_ _ _ _ [i.e., 8]	**0 1 1 1 1 1 1 1** [i.e., 127]	**0 1 1 1 1 1 1 1 1** [i.e., 255]
_ _ _ [i.e., 5]	_ _ _ _ [i.e., 9]	_ _ _ _ _ _ _ _ [i.e., 128]	_ _ _ _ _ _ _ _ [i.e., 256]

<u>Question 6:</u> Explain/Discuss the following concepts in detail *(Search the internet to include more information & details)***.**

1. The properties (characteristics) of an Algorithm.
2. Algorithmic Efficiency (the efficiency of an algorithm).
3. The five computer generations and the key characteristics of computers in each generation.
4. The classification of computer systems (i.e., the six basic categories of computers).
5. How images are represented in a computer system and the relationship between videos and images.
6. Artificial Intelligence, and mention some of its applications in everyday life.
7. Software Engineering.
8. The three generic types of errors in a computer program.

<u>Question 7:</u> Watch the following video that introduces "Apple Vision Pro," and then answer the following questions [https://www.youtube.com/watch?v=TX9qSaGXFyg]:

1. *"Your photos can be life-size or any size, so your living room becomes a gallery. And panoramas wrap around you as if you're right where you took them."* This functionality is an application of: **(a)** Digital Image Processing. **(b)** Computer Graphics. **(c)** Computer Vision. **(d)** None of the above.

2. *"Apple Vision Pro is Apple's first-ever 3D camera. Now you can capture photos and videos with remarkable depth."* This feature is an example of ____ devices. **(a)** Storage. **(b)** Communication. **(c)** Output. **(d)** Input.
3. *"Since your eyes see the world with incredible resolution, we built a micro-OLED display system that fits 64 pixels in the same amount of space as a single iPhone pixel and packs 23 million into two panels the size of a postage stamp. That's more than a 4K TV for each eye, giving you jaw-dropping, life-like clarity."* This feature is an example of _____ devices. **(a)** Storage. **(b)** Communication. **(c)** Output. **(d)** Input.
4. *"Experiences on Vision Pro can also expand in three dimensions, filling the entirety of your space. Like in the Mindfulness app, where you can create a moment of calm."* This application can be considered as an example of: **(a)** Bio-Informatics. **(b)** Psycho-Informatics. **(c)** Eco-Informatics. **(d)** Engineering Informatics.
5. *"To power a spatial computer like Apple Vision Pro required an innovative dual-chip design. M2 provides phenomenal performance, and a brand-new chip, R1, processes sensor data at incredible speed, virtually eliminating lag, so experiences take place in real-time right in front of your eyes."* Both M2 and R1 are examples of _____ devices. **(a)** Storage. **(b)** Communication. **(c)** Output. **(d)** Input. **(e)** Processing.
6. *"Spatial Audio surrounds you and makes you feel like you're a part of the action."* This is mainly an application of: **(a)** Databases. **(b)** Data Structures. **(c)** Digital Signal Processing. **(d)** Computer Vision.
7. *"And Safari expands so you can see all your open tabs."* Safari is considered a: (a) System Software. (b) General-purpose Application. (c) Special-Purpose Application. (d) Utility Program. (e) Library Program.
8. *"This is visionOS, Apple's first-ever spatial operating system. It's familiar yet groundbreaking. You navigate with your eyes. Simply tap to select, flick to scroll, and use your voice to dictate."* VisionOS is considered a: (a) System Software. (b) General-purpose Application. (c) Special-Purpose Application. (d) Utility Program. (e) Library Program.
9. *"Your Persona dynamically reflects your face and hand movements, so when you're chatting, people see your eyes, hands, and true expressions."* Creating a persona/avatar that captures and reflects your movements and expressions is an application of both Computer Vision and Computer Graphics. (a) True. (b) False.
10. *"We engineered a system that uses advanced machine learning to represent you realistically when you're in FaceTime."* Machine learning is a subset/subfield of AI that focuses on the development of algorithms and statistical models that allow computers to learn and make predictions or decisions without being explicitly programmed. (a) True. (b) False.
11. *"The web comes to life at a fantastic scale. Text is crisp and easy to read. Browsing the internet feels new."* These features are all aspects that ____ specialists and designers may work on to optimise the user's experience. (a) Databases. (b) Programming Languages. (c) Human-Computer Interaction. (d) Artificial Intelligence. (e) Computer Vision.

With my best wishes;
Dr. Amr S. Ghoneim

Exercises – Problems Sheet # 2: Pseudocodes, Flowcharts, & an Intro. to C

No. Of Questions: 5 **No. Of Pages: 6**

- **To be submitted during the Labs of week 5.**
- **Students will lose 2 marks if this homework is not delivered on time or found to be copied.**
- **The submitted solutions should be handwritten and NOT typed/printed.**

<u>Question 1:</u> Briefly describe – in your own words – the following terms:

1. Implementation.
2. Machine Code (Machine Language).
3. An Assembler.
4. The *main* function.
5. The *printf* function.
6. Escape Sequences *(give 2 examples)*.
7. Preprocessor Directives *(give an example)*.
8. A header file.
9. Prompt Messages.
10. Conversion/Format Specifier.
11. The *return* Keyword.
12. A Program's Readability.
13. The datatype *void*.
14. A String.

<u>Question 2:</u> Briefly differentiate between the following terms:

1. The <stdio.h> header file versus the <math.h> header file.
2. Built-in functions versus User-Defined functions.
3. A Function Header versus a Function Body.
4. Single-Lined Comments versus Multiple-Lined Comments.
5. High-Level Language versus a Low-Level Language.

<u>Question 3:</u> Given the following <u>solved example</u> for tracing an algorithm (represented as a pseudocode):

Pseudocode 0: Calculate and display the factorial of a +ve integer using a loop.

```
1. Input n // Assume the user input 5
2. Set factorial = 1
3. For i from 1 to n DO
4.    Set factorial = factorial * i
5. End For
6. Print factorial
7. End
```

Step #	n	i	factorial	Display (Screen)
1	5	-	-	-
2	5	-	1	-
3	5	1	1	-
4	5	1	1	-
3	5	2	1	-
4	5	2	2	-
3	5	3	2	-
4	5	3	6	-
3	5	4	6	-
4	5	4	24	-
3	5	5	24	-
3	5	5	120	120

Trace (and determine the output) of the following 4 pseudocodes by completing the tables:

Pseudocode 1: Calculate and display the sum of two given numbers.

```
1. Prompt "Please input the first number: "
2. Input num1 // Assume the user inputs 11
3. Prompt "Please input the second number: "
4. Input num2 // Assume the user inputs 22
5. Set sum = num1 + num2
6. Print "The sum of the two numbers is: ", sum
7. End
```

Step #	num1	num2	sum	Display (Screen)
1				
2				
3				
4				
5				
6				

Pseudocode 2: Determine whether a given positive integer is a prime number.

```
1. INPUT num
2. IF num < 2 THEN
3.    PRINT "Not Prime"
4.    END
5. END IF
6. FOR i from 2 to sqrt(num) DO // sqrt(num) is the square root of num
7.    IF num mod i == 0 THEN // mod is "modulo"/"modulus", num mod i ≡ num % i
8.          PRINT "Not Prime"
9.          END
10.   END IF
11. END FOR
12. PRINT “Prime”
13. END
```

A Solution (a solved example assuming num = 1):

Step #	Num	num < 2	i	sqrt(num)	num mod i	num mod i == 0	Display (Screen)
1	**1**	-	-	-	-	-	-
2	1	**true**	-	-	-	-	-
3	1	**true**	-	-	-	-	**Not Prime**
4	**END: Terminate the program**						

Step #	num	num < 2	i	sqrt(num)	num mod i	num mod i == 0	Display (Screen)
1	**25**	-	-	-	-	-	-
2							
6							
7							
6							
7							
6							
7							
6							
7							
8							
9							

Pseudocode 3: Approximate Pi (π) [using the Gregory-Leibniz series]

```
 1. Input: The number of terms 'n' to use in the approximation // Note: The more
    terms you use (higher 'n'), the more accurate the approximation will be.
 2. Output: An approximation of π
 3. Initialize pi = 0
 4. Initialize sign = 1
 5. Initialize denominator = 1
 6. For i from 1 to n do
 7.   pi = pi + sign * 4 / denominator
 8.   denominator = denominator + 2
 9.   sign = sign * -1
10. End For
11. Print Approximation of π after n terms is: pi
```

Step #	n	pi	sign	denominator	i	Display (Screen)
1	**8**	-	-	-	-	-
2						
3						
4						
5						
6						
7						
8						
9						
6						
7						
8						
9						
6						
7						
8						
9						
6						
7						
8						
9						
6						
7						
8						
9						
6						
7						
8						
9						
6						
7						
8						
9						
6						
7						
8						
9						
10						

Pseudocode 4: Display the Fibonacci series up to the nth term using loops.

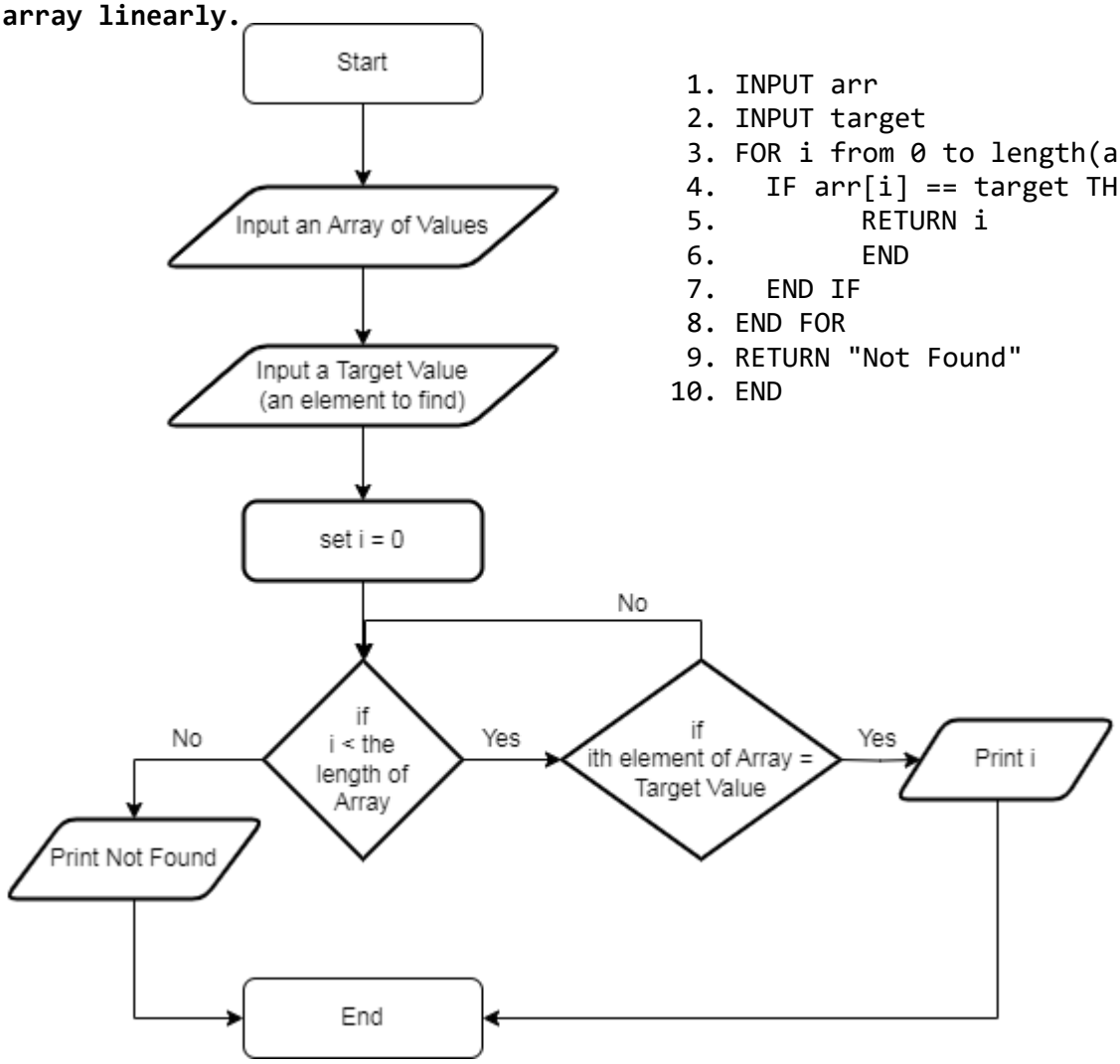
```
 1. Prompt "Please input a positive integer: "
 2. Input: A positive integer 'n' // Assume the user inputs 7
 3. Output: The first 'n' terms of the Fibonacci series
 4. Initialize first = 0
 5. Initialize second = 1
 6. Print first
 7. Print second
 8. For i from 3 to n do
 9.   next_term = first + second
10.   Print next_term
11.   first = second
12.   second = next_term
13. End For
```

Step #	n	First	second	i	next_term	Display (Screen)
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
8						
9						
10						
11						
12						
8						
9						
10						
11						
12						
8						
9						
10						
11						
12						
8						
9						
10						
11						
12						

<u>Question 4:</u> Draw a suitable block diagram explaining the basic steps for compiling a program, then briefly describe them (i.e., these steps).

<u>Question 5:</u> Given the following <u>solved example</u> for representing an algorithm using both a pseudocode and a flowchart:

Pseudocode 0: Write down an Algorithm using pseudocode (and represent it graphically using a flowchart) to perform a Linear Search; that is, to find a specific element (a target value) in an array by checking each element in the array linearly.




```
1. INPUT arr
2. INPUT target
3. FOR i from 0 to length(arr) - 1 DO
4.   IF arr[i] == target THEN
5.         RETURN i
6.         END
7.   END IF
8. END FOR
9. RETURN "Not Found"
10. END
```

For each of the following, write down an algorithm using pseudocode, and represent it graphically (i.e., visually) using a flowchart:

1) **Hello, World**: Write an algorithm for a program that displays "Hello, World!" on the screen.

2) **Sum of Two Numbers**: Create an algorithm for a program that adds two numbers provided by the user and displays their sum.
   [***Hint**: Prompt the user for input, perform the addition, and display the result.*]

3) **Calculate Area**: Write an algorithm for a program that calculates and displays the area of a rectangle with user-provided length and width.
   [***Equation**: Area = Length x Width.*]
   [***Hint**: Prompt the user for the length and width, then calculate and display the area.*]

4) **Count to N**: Write an algorithm for a program that counts from 1 to N, where N is a user-entered number.
[***Hint**: Use a loop to count from 1 to N and display the numbers.*]

5) **Even or Odd**: Create an algorithm for a program that checks if a user-entered number is even or odd.
[***Equation**: Number % 2.*]
[***Hint**: Use the modulo operator to check if the number is even or odd.*]

6) **Fibonacci Sequence**: Write an algorithm for a program that displays the first N numbers in the Fibonacci sequence.
[***Equation**: Fibonacci(n) = Fibonacci(n-1) + Fibonacci(n-2).*]
[***Hint**: Use a loop – or recursion – to generate the Fibonacci sequence.*]

7) **Factorial Calculation**: Create an algorithm for a program that calculates the factorial of a user-entered number.
[***Hint**: Use a loop – or recursion – to calculate the Factorial.*]

8) **Prime Number Checker**: Write an algorithm for a program that checks if a number provided by the user is prime.
*[**Hint**: By definition, a prime number is a positive integer greater than 1 with exactly two distinct positive divisors: 1 and itself. Thus, 1 is not considered a prime number, since 1 only has one distinct positive divisor (itself); it does not meet the criteria for being a prime number. Prime numbers start from 2 and go up from there.]*

9) **Temperature Converter**: Create an algorithm for a program to convert temperatures between Celsius and Fahrenheit.
[***Equations**:*
Celsius to Fahrenheit: F = (C x 9/5) + 32.
Fahrenheit to Celsius: C = (F - 32) x 5/9.]
[***Hint**: Depending on the user's choice, apply the appropriate equation for conversion.*]

10) **Simple Calculator**: Develop an algorithm for a basic calculator program that can perform addition, subtraction, multiplication, and division.
[***Hint**: Use if-else statements to perform different operations.*]

11) **Find Maximum Number**: Create an algorithm for a program that finds the largest number among three user-entered numbers.
[***Hint**: Use conditional statements to compare the numbers.*]

12) **Leap Year Checker**: Write an algorithm for a program that checks if a year entered by the user is a leap year.
[***Equations**: Leap year if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0).*]

13) **Guess the Number**: Develop an algorithm for a game where the computer generates a random number, and the user tries to guess it.

With my best wishes;
Dr. Amr S. Ghoneim

Exercises – Problems Sheet # 3: Basic Programming Concepts in C

No. Of Questions: 4 **No. Of Pages: 2**

- **To be submitted during the Labs of week 6.**
- **Students will lose 2 marks if this homework is not delivered on time or found to be copied.**
- **The submitted solutions should be handwritten and NOT typed/printed.**

Question 1: You are tasked with creating a simple C program that functions as a basic calculator. The program should provide a menu of four operations: Addition, Subtraction, Multiplication, and Division. Users will be prompted to choose an operation, enter two numbers, and then receive the result with a tax rate applied. Your program should adhere to the following program requirements and provide a user-friendly calculator interface.

1. **Development Environment**: Use a C development environment, such as Code::Blocks.
2. **Constant Variable**: Define a constant variable named TAX_RATE with a value of 0.08 to represent the tax rate.
3. **Macro for Maximum Operations**: Define a macro named MAX_OPERATIONS with a value of 4 to specify the maximum number of available operations.
4. **Variable Declarations**: Declare the following variables of different data types:
    a. float num1 to store the first number.
    b. float num2 to store the second number.
    c. int choice to store the user's choice of operation.
5. **Menu Display**: Display a menu of operations as follows:
    Calculator Menu:
    1. Addition
    2. Subtraction
    3. Multiplication
    4. Division
6. **Choice Input**: Prompt the user to choose an operation and read their choice. Ensure that the choice is within the range of 1 to MAX_OPERATIONS.
7. **Number Input**: Prompt the user to enter two floating-point numbers and read them.
8. **Operation Execution**: Perform the selected operation based on the user's choice using only conditional expressions *(without using if statements, switch statements, or conditional/ternary operators)*.
9. **Error Handling**: Ensure that division is only performed if the second number (num2) is not zero.
10. **Tax Calculation**: Apply the tax rate (stored in TAX_RATE) to the result of the operation.
11. **Result Display**: Display the final result with exactly two decimal places.
12. **Program Exit**: Exit the program with a success code of 0.

Question 2: What is the output of the following program? Justify your choice.

```
#include <stdio.h>
int main() {
    int length = 25, width = 60, height;
    if (length = 50)
       height = 4;
    else
       height = 8;
       printf ("Length = %d, Width = %d, and Height = %d.", length, width, height);
    return 0;
}
```

a) The program will display: Length = 25, Width = 60, and Height = 8.
b) The program will display: Length = 25, Width = 60, and Height = 4.
c) The program will display: Length = 50, Width = 60, and Height = 8.
d) The program will display: Length = 50, Width = 60, and Height = 4.
e) The program won't run (it will result in a syntax/compilation error).
f) The program will run, but it will print nothing since "length = 50" is an assignment statement (not an equality) that will result in true. Therefore, the printf statement will not be executed because it belongs to the else block.

Question 3: For the given arithmetic expressions, determine the order of precedence of the arithmetic operations. Clearly provide the sequence in which the arithmetic operations should be performed within each expression to illustrate the order of precedence.

a) 2 * 3 + 2 * (3 * 4 + 5 * 6)
b) 8 - 4 + (6 + 9 / (3 * 2))
c) 5 * (7 + 4 * (8 + 2)) / 6 * (9 + 3)

Question 4: Write a Program in C for each of the following requirements:

1) **Hello, World**: Write a C program that displays "Hello, World!" on the screen.
2) **Sum of Two Numbers**: Write a C program that adds two numbers provided by the user and displays their sum.
3) **Calculate Area**: Write a C program that calculates and displays the area of a rectangle with user-provided length and width.
4) **Temperature Converter**: Write a C program to convert temperatures from Celsius to Fahrenheit. [***Equation**: Celsius to Fahrenheit: F = (C x 9/5) + 32.*]
5) **Swap Two Numbers** (**using a temporary variable**): Write a C program that swaps two numbers provided by the user using a temporary variable.
6) **Swap Two Numbers** (**without a temporary variable**): Write a C program that swaps two numbers provided by the user without using a temporary variable.
7) **Find the Maximum Number** (**without using** if **statements,** switch **statements, or conditional/ternary operators**): Write a C program that finds the largest number among three user-entered numbers without using if statements, switch statements, or conditional/ternary operators anywhere in the program. [***Hint**: Use a combination of arithmetic operations and logical expressions.*]

With my best wishes;
Dr. Amr S. Ghoneim

Exercises – Problems Sheet # 4: Bitwise Operations & Selection Control in C

No. Of Questions: 3 **No. Of Pages: 2**

- **To be submitted during the Labs of weeks 8 or 9.**
- **Students will lose 2 marks if this homework is not delivered on time or found to be copied.**
- **The submitted solutions should be handwritten and NOT typed/printed.**

Question 1: State whether the following statement is true or false. If it is false, explain why.

The following code should print whether a given integer is odd or even:

```
switch ( value % 2 )
{
      case 0:
            printf( "An even integer\n" );
      case 1:
            printf( "An odd integer\n" );
}
```

Question 2: Write a Program in C for each of the following requirements:

1) **Even or Odd** (**using an if statement**)**:** Write a C program that checks if a user-entered number is even or odd, using an if statement.

2) **Even or Odd** (**using a conditional/ternary operator**)**:** Write a C program that checks if a user-entered number is even or odd, using a conditional (ternary) operator.

3) **Leap Year Checker** (**using an if statement**): Write the C code for a program that checks if a year entered by the user is a leap year using an if statement.
[***Equations****: Leap year if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0).*]

4) **Leap Year Checker** (**using a conditional/ternary operator**): Write the C code for a program that checks if a year entered by the user is a leap year, using a conditional/ternary operator.

5) **Find the Maximum Number** (**using if–else statements**): Write a C program that finds the largest number among three user-entered numbers, using if–else statements.

6) **Find the Maximum Number** (**using a conditional/ternary operator**): Write a C program that finds the largest number among three user-entered numbers using a conditional (ternary) operator.

7) **Simple Calculator** (**using if–else statements**): Write the C code for a basic calculator program that can perform addition, subtraction, multiplication, modulo, and division.
[***Hint****: Use if-else statements to perform the different operations.*]

8) **Simple Calculator** (**using a switch–case statement**): Write the C code for a basic calculator program that can perform addition, subtraction, multiplication, modulo, and division.
[***Hint****: Use a switch-case statement to perform the different operations.*]

9) **Determine the Letter Grade** (**using a switch–case statement**): Write the C code for a basic program that takes a mark (from 0 to 100) as input and provides a corresponding letter grade ("Excellent," "Very Good," "Good," "Pass," or "Fail" for the following ranges respectively: 85 to 100, 75 to 84, 65 to 74, 50 to 64, and 0 to 49).
[***Hint****: Use a range in the switch-case statement*]

<u>Question 3</u>: Trace the following program (while trying different values for both variables a and b). What does this code do?

```
#include <stdio.h>
int main()
{
    unsigned short int a, b;

    printf("Please enter 2 integer values:\n");
    printf("a = "); scanf("%hu", &a);
    printf("b = "); scanf("%hu", &b);

    a = a ^ b;
    b = a ^ b;
    a = a ^ b;

    // Display updated values
    printf("Updated values:\n");
    printf("a = %d\n", a);
    printf("b = %d\n", b);

    return 0;
}
```

With my best wishes;
Dr. Amr S. Ghoneim

Exercises – Problems Sheet # 5: Repetition Control (& more on Selection Control) in C

No. Of Questions: 1 **No. Of Pages: 6**

- **To be submitted during the Labs of weeks 9 or 10.**
- **Students will lose 2 marks if this homework is not delivered on time or found to be copied.**
- **The submitted solutions should be handwritten and NOT typed/printed.**

Question 1: Answer the following:

1) Write a C program that uses a '**for**' loop to display numbers from 1 to 10.

```
1  2  3  4  5  6  7  8  9  10
```

2) Create a C program that utilises a '**while**' loop to print even numbers from 2 to 20.

```
2  4  6  8  10  12  14  16  18  20
```

3) Develop a C program with a '**do while**' loop to calculate the sum of numbers from 1 to 50.

```
Sum: 1275
```

4) Write a C program using a '**for**' loop to find the factorial of a given number.

```
Enter a number: 5
Factorial: 120
```

5) Create a C program that uses a '**while**' loop to prompt the user for numbers until they enter 0. Display the sum of these numbers. [***Hint****: use an infinite loop and '****break****' in your program.*]

```
Enter a number (enter 0 to stop): 3
Enter a number (enter 0 to stop): 7
Enter a number (enter 0 to stop): 2
Enter a number (enter 0 to stop): 0
Sum: 12
```

6) Develop a C program with a '**do while**' loop to find the average of a set of positive numbers entered by the user. The loop should terminate when a negative number is encountered.

```
Enter positive numbers (enter a negative number to stop): 8
Enter positive numbers (enter a negative number to stop): 4
Enter positive numbers (enter a negative number to stop): 9
Enter positive numbers (enter a negative number to stop): -1
Average: 7.0
```

7) Write a C program that uses a '**for**' loop to print the multiplication table of a given number.

```
Enter a number: 7
Multiplication Table for 7:
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```

8) Create a C program using a '**while**' loop to display the prime numbers between 1 and 50.

```
Prime numbers between 1 and 50:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
```

9) Develop a C program with a '**do while**' loop to reverse a given number.

```
Enter a number: 12345
Reversed number: 54321
```

10) Write a C program that uses a '**for**' loop to print a pattern of stars:

```
*
**
***
****
```

11) Write a C program using a '**while**' loop to check whether a given number is a palindrome or not.

```
Enter a number: 121
121 is a palindrome.
```

12) Write a C program that uses a '**for**' loop to generate the Fibonacci series up to a certain term.

```
Enter the number of terms in the Fibonacci series: 8
Fibonacci Series up to 8 terms:
0, 1, 1, 2, 3, 5, 8, 13,
```

13) Create a C program using a '**while**' loop to find the Greatest Common Divisor of two numbers.

```
Enter the first number: 48
Enter the second number: 18
The Greatest Common Divisor (GCD) is: 6
```

14) Develop a C program with a '**do while**' loop to print a menu and perform different operations based on the user's choice (e.g., add, subtract, multiply, divide).

```
Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
0. Exit
Enter your choice: 1
Enter two numbers to add: 5 3
Result: 8.00

Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
0. Exit
Enter your choice: 3
Enter two numbers to multiply: 4 2
Result: 8.00

Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
0. Exit
Enter your choice: 4
Enter two numbers to divide: 10 2
Result: 5.00

Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
0. Exit
Enter your choice: 0
Exiting the program. Goodbye!
```

15) Write a C program that uses a '**for**' loop to print a triangle pattern of numbers:

```
1
22
333
4444
```

16) Create a C program using a '**while**' loop to implement a guessing game. The program should generate a random number, and the user has to guess it.

```
Welcome to the Guessing Game!
Enter your guess (between 1 and 100): 50
Too low. Try again.
Enter your guess (between 1 and 100): 75
Too high. Try again.
Enter your guess (between 1 and 100): 60
Too low. Try again.
Enter your guess (between 1 and 100): 70
Too high. Try again.
Enter your guess (between 1 and 100): 65
Congratulations! You guessed the correct number (65) in 5 attempts.
```

17) Write a C program that uses a '**for**' loop to print the ASCII values and corresponding characters from 65 to 90.

```
ASCII value: 65, Character: A
ASCII value: 66, Character: B
ASCII value: 67, Character: C
ASCII value: 68, Character: D
ASCII value: 69, Character: E
ASCII value: 70, Character: F
ASCII value: 71, Character: G
..
..
..
ASCII value: 84, Character: T
ASCII value: 85, Character: U
ASCII value: 86, Character: V
ASCII value: 87, Character: W
ASCII value: 88, Character: X
ASCII value: 89, Character: Y
ASCII value: 90, Character: Z
```

18) Create a C program using a '**while**' loop to find the sum of digits of a given number.

```
Enter an integer: 12345
The sum of digits of 12345 is: 15
```

19) Develop a C program with a **nested** '**for**' loop to print a rectangular pattern of stars:

```
*****
*****
*****
```

20) Write a C program using **nested loops** to print a multiplication table for numbers 1 to 5.

```
1  2   3   4   5
2  4   6   8   10
3  6   9   12  15
4  8   12  16  20
5  10  15  20  25
```

21) Write a C program using a '**for**' loop and the **continue** keyword to print the odd numbers between 1 and 10.

```
1  3  5  7  9
```

22) Write a C program using a **nested** '**while**' **loop** and **switch-case statements** to print a menu-driven calculator. The program should also utilise the keywords **break**, **continue**, and **default**.

- The program is a simple calculator that allows users to perform basic arithmetic operations such as addition, subtraction, multiplication, and division. It utilises a menu-driven approach, prompting the user to select an operation and input two numbers for the calculation. The program continues to run until the user chooses to exit.
- The user is prompted to enter a choice corresponding to a specific arithmetic operation. The valid choices are 1, 2, 3, 4, and 0 (for exit).
- If the user chooses an arithmetic operation (1 to 4), they are required to enter two integer values for the calculation.
- For division (choice 4), the program checks if the denominator is not zero to avoid division by zero errors. If the user attempts division with a denominator of zero, an error message is displayed, and the user is prompted to enter the denominator again.
- Invalid choices for arithmetic operations result in an error message, and the user is prompted to enter a valid choice.
- Upon selecting an operation and providing the required input, the program prints the result of the operation.
- If the user chooses to exit (entering 0), the program prints a farewell message and terminates.
- The program uses a while (1) loop to allow continuous operation until the user chooses to exit.
- The break statement is used to exit the loop when the user selects 0 to exit the program.

```
Calculator Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
0. Exit
Enter your choice: 1
Enter two numbers: 5 3
Result: 8

Calculator Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
0. Exit
Enter your choice: 4
Enter two numbers: 15 0
Error: Division by zero is not allowed.

Calculator Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
0. Exit
Enter your choice: 5
Invalid choice. Please enter a number between 0 and 4.

Calculator Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
0. Exit
Enter your choice: 0
Exiting the calculator. Goodbye!
```

With my best wishes;
Dr. Amr S. Ghoneim

Exercises – Problems Sheet # 6: Functions & Recursion using C

No. Of Questions: 1 **No. Of Pages: 4**

- **To be submitted during the Labs of weeks 10 or 11.**
- **Students will lose 2 marks if this homework is not delivered on time or found to be copied.**
- **The submitted solutions should be handwritten and NOT typed/printed.**

Question 1: Answer the following:

1) Write a C program that defines a function '**greet**' to print a greeting message. The function should not accept any arguments.

```
Hello, welcome to the program!
```

2) Write a C program that defines a function '**add**' to add two numbers. The function should accept two integer arguments.

```
Sum of 5 and 7: 12
```

3) Write a C program that defines a function '**calculateArea**' to calculate the area of a circle.

```
Area of the circle with radius 2.50 is: 19.63
```

4) Write a C program that defines a function '**multiply**' to multiply three numbers. The function should accept three floating-point arguments.

```
Product of 2.5, 3.0, and 1.5 of : 11.25
```

5) Write a C program that defines a function '**isEven**' to check if a given integer is even. The function should accept one integer argument and return a Boolean value.

```
6 is even.
7 is odd.
```

6) Write a C program that defines a function '**power**' to calculate the power of a number. The function should accept two arguments, the base (integer) and the exponent (integer), and return the result.

```
2^3 = 8
```

7) Write a C program that defines a function '**average**' to calculate the average of three numbers. The function should accept three floating-point arguments and return the average.

```
Average of 4.5, 3.0, and 6.5: 4.33
```

8) Write a C program that defines a function '**max**' to find the maximum of two numbers. The function should accept two integer arguments and return the maximum.

```
Maximum of 8 and 5: 8
```

9) Write a C program that defines a function '**sumOfSquares**' to calculate the sum of squares of two numbers. The function should accept two arguments (integers) and return the sum of their squares.

```
Sum of squares of 3 and 4: 25
```

10) Write a C program that defines a function '**printPattern**' to print a pattern of stars. The function should not accept any arguments and should print a rectangle of stars (e.g., 5x3).

```
* * *
* * *
* * *
* * *
* * *
```

11) Write a C program that defines a function '**isPrime**' to check if a given number is prime. The function should accept one integer argument and return a Boolean value.

```
13 is prime.
```

12) Write a C program that defines a function '**printMenu**' to print a menu for a calculator program. The function should not accept any arguments.

```
Calculator Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
0. Exit
```

13) Write a C program that defines a function '**printTable**' to print the multiplication table of a given number. The function should accept one integer argument.

```
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

14) Write a C program that defines a function '**printPatternWithSize**' to print a pattern of stars with a given number of rows and columns. The function should accept two integer arguments.

```
Printing Pattern with Size 4 by 6:
* * * * * *
* * * * * *
* * * * * *
* * * * * *
```

15) Write a C program that defines a function '**reverseNumber**' to reverse the digits of a given integer. The function should accept one integer argument and return the reversed number.

```
Original number: 12345
Reversed number: 54321
```

16) Write a C program that defines a function '**gcd**' to find the GCD (Greatest Common Divisor) of two numbers using Euclid's algorithm.

```
GCD of 24 and 36 is: 12
```

17) Write a C program that defines a function '**factorial**' to calculate the factorial of a number using recursion. The function should accept one integer argument and return the factorial.

```
Factorial of 5: 120
```

18) Write a C program that defines a function '**powerRecursive**' to calculate the power of a number using recursion. The function should accept two arguments, the base (integer) and the exponent (integer), and return the result.

```
2^3 = 8
```

19) Write a C program that defines a function '**fibonacci**' to generate the first N terms of the Fibonacci sequence using recursion. The function should accept one integer argument (N) and print the sequence.

```
Fibonacci sequence (first 7 terms): 0 1 1 2 3 5 8
```

20) Write a C program that defines a function '**swap**' to swap the values of two variables. The function should accept two integer arguments and swap their values. [***Hint**: This is a C++ program.*]

```
Before swapping: x = 5, y = 8
After swapping: x = 8, y = 5
```

21) Write a C program that defines a function '**calculateSumAndAverage**' to calculate the sum and average of two numbers. The function should accept two integer arguments and return the sum and average using pass by reference. [***Hint**: This is a C++ program.*]

```
The Sum of 5 and 8: 13
The Average of 5 and 8: 6.50
```

22) Research the keyword '**static**' in C, the trace the following code. What is the effect of using '**static**'?

```c
#include <stdio.h>

// Function prototype
void fibonacci(int N);

int main() {
    int terms = 7;
    printf("Fibonacci sequence (first %d terms): ", terms);

    // Call the fibonacci function
    fibonacci(terms);

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

// Function definition
void fibonacci(int N) {
    static int a = 0, b = 1, sum;

    if (N > 0) {
        sum = a + b;
        printf("%d ", sum);
        a = b;
        b = sum;
        fibonacci(N - 1);
    }
}
```

With my best wishes;
Dr. Amr S. Ghoneim

Exercises – Problems Sheet # 7: Number Systems

No. Of Questions: 13 **No. Of Pages: 2**

- **To be submitted during the Labs of week 12.**
- **Students will lose 2 marks if this homework is not delivered on time or found to be copied.**
- **The submitted solutions should be handwritten and NOT typed/printed.**

Question 1: Convert the following decimal numbers to binary:

a) 164	b) 255	c) 94	d) 68.5625	e) 43.625

Question 2: Redo question 1 but convert the decimal numbers to hexadecimal.

Question 3: Redo question 1 but convert the decimal numbers to octal.

Question 4: Convert the following binary numbers to decimal:

a) 1010	b) 11001	c) 10000	d) 110111.110	e) 10010101.111

Question 5: Convert the following binary numbers to octal:

a) 10110101	c) 10101001101	e) 01101011100.01100
b) 110011111	d) 1110110.101	

Question 6: Redo question 5 but convert the binary numbers to hexadecimal.

Question 7: Convert the following Hexadecimal numbers to octal:

a) 4E8	b) 2F1	c) D78	d) 5C0	e) 5BCA

Question 8: Perform the binary addition of:

a) 11 + 11	c) 10001 + 1111	e) 11111 + 11111
b) 1001 + 110	d) 111001 + 11111	

Question 9: Perform the binary subtraction of:

a) 10011 – 10001	c) 10001 – 1111	e) 1001 – 110
b) 101000 – 1001	d) 111001 – 11111	

Question 10: Derive the truth tables for the following expressions:

a) NOT (A OR B)

b) A OR (B AND C)

c) (NOT A) OR B

d) (A NAND B) AND C

e) (NOT A) AND (A OR (NOT B))

Question 11: Find the 1's and 2's complement of the following:

a) 1011 1011

b) 0101 1000

c) 0110 1101 0111

d) 1011 0110 1101

e) 0001 0110 1010

Question 12: Write a C program that reads a decimal number and a target base (either 8 for octal or 16 for hexadecimal) as input from the user. Implement a function '**convertToBase**' that converts the given decimal number to the specified target base and prints the result. If the target base is neither 8 nor 16, print "Unsupported target base." Use format specifiers to print the number in the target base.

Question 13: Trace the following C program and explain what it does.

```
#include <stdio.h>
int x(int a, int b) {
    if (b == 0) { return a; }
    int sum = a ^ b;
    int carry = (a & b) << 1;
    return x(sum, carry);
}
int main() {
    int num1, num2;
    printf("Enter the first integer: "); scanf("%d", &num1);
    printf("Enter the second integer: "); scanf("%d", &num2);
    int result = x(num1, num2); printf("%d\n", result);
    return 0;
}
```

With my best wishes;

Dr. Amr S. Ghoneim