**/\***

**Name: nwosu emmanuel.**

**Department: Mobile App Development**

**EXAM II**

**\*/**

void main()

{

**// 1. Define a Dart function that takes two parameters (int a, int b) and returns their sum.**

  int A = 100;

  int B = 30;

  int result = A + B;

  print("1. The sum is= $result");

}

**// 2. Write a function that calculates the factorial of a given number.**

void main() {

int number = 4;

int factorial = calculateFactorial(number);

print('The factorial of $number is: $factorial');

}

int calculateFactorial(int number)

{

int result = 1;

for (int i = 1; i <= number; i++)

{

result \*= i;

}

return result;

}

**// 3. Create a function that checks if a number is prime.**

void main()

{

int number = 2;

print('${isPrime(number) ? "$number is" : "$number is not"} a prime number.');

}

bool isPrime(int number)

{

if (number < 2) return false;

for (int i = 2; i <= (number / 2); i++)

{

if (number % i == 0) return false;

}

return true;

}

**// 4. Implement a function that converts Celsius to Fahrenheit.**

void main() {

double celsiusTemperature = 25.0;

double fahrenheitTemperature = celsiusToFahrenheit(celsiusTemperature);

print('$celsiusTemperature degrees Celsius is $fahrenheitTemperature degrees Fahrenheit.');

}

double celsiusToFahrenheit(double celsius) {

return (celsius \* 9 / 5) + 32;

}

**// 5. Write a Dart function that takes a string as input and returns the reversed string.**

void main()

{

String inputString = "I .love SID";

String reversedString = reverseString(inputString);

print('Original String: $inputString');

print('Reversed String: $reversedString');

}

String reverseString(String input) {

List<String> characters = input.split('');

characters = characters.reversed.toList();

return characters.join('');

}

/**/ 6. Create a class named Person with attributes name and age. Implement a method to display the person's information.**

class Person

{

String name;

int age;

Person(this.name, this.age); // Constructor

void displayInfo() // Method to display person's information

{

print('Name: $name, Age: $age');

}

}

void main()

{

Person person1 = Person('David', 30); // Creating an instance of the Person class

person1.displayInfo(); // Calling the displayInfo method to show person's information

}

**// 7. Create a class named Person with attributes name and age. Implement a method to display the person's information.**

class Rectangle

{

double length;

double width;

Rectangle(this.length, this.width); // Constructor

// Method to calculate the area

double calculateArea()

{

return length \* width;

}

}

void main()

{

Rectangle myRectangle = Rectangle(5.0, 3.0); // Creating an instance of the Rectangle class

double area = myRectangle.calculateArea(); // Calculating and printing the area of the rectangle

print('Area of the rectangle: $area square units');

}

**// 8. Implement a class Circle with a method to calculate the circumference.**

import 'dart:math';

class Circle

{

double radius;

Circle(this.radius); // Constructor

// Method to calculate the circumference

double calculateCircumference()

{

return 2 \* pi \* radius;

}

}

void main()

{

Circle myCircle = Circle(5.0); // Creating an instance of the Circle class

// Calculating and printing the circumference of the circle

double circumference = myCircle.calculateCircumference();

print('Circumference of the circle: $circumference units');

}

**// 9. Create a class BankAccount with methods to deposit and withdraw money, and to check the account balance.**

class BankAccount

{

String accountHolder;

double balance;

BankAccount(this.accountHolder, this.balance); // Constructor

// Method to deposit money

void deposit(double amount)

{

if (amount > 0)

{

balance += amount;

print('Deposit successful. New balance: ₦${balance.toStringAsFixed(2)}');

}

else

{

print('Invalid deposit amount.');

}

}

// Method to withdraw money

void withdraw(double amount)

{

if (amount > 0 && amount <= balance)

{

balance -= amount;

print('Withdrawal successful. New balance: ₦${balance.toStringAsFixed(2)}');

}

else

{

print('Invalid withdrawal amount or insufficient funds.');

}

}

// Method to check account balance

void checkBalance()

{

print('Account balance for $accountHolder: ₦${balance.toStringAsFixed(2)}');

}

}

void main() {

// Creating an instance of the BankAccount class

BankAccount myAccount = BankAccount('Nzeamalu David C.', 1000.0);

// Depositing and checking balance

myAccount.deposit(500.0);

myAccount.checkBalance();

// Withdrawing and checking balance

myAccount.withdraw(200.0);

myAccount.checkBalance();

}

**10. Write a class Student with attributes name, age, and grade. Implement a method to determine if the student passed or failed.**

class Student

{

String name;

int age;

int grade;

// Constructor to initialize attributes

Student(this.name, this.age, this.grade);

// Method to check if the student passed or failed

void checkPassOrFail()

{

final passingGradeThreshold = 40;

if (grade >= passingGradeThreshold)

{

print('$name has passed the exam with a grade of $grade.');

} else {

print('$name has failed the exam with a grade of $grade.');

}

}

}

void main() {

// Creating an instance of the Student class

Student student1 = Student('Nzeamalu David C.', 18, 75);

// Checking if the student passed or failed

student1.checkPassOrFail();

}

**// 11. Define a class Calculator with methods for basic arithmetic operations (add, subtract, multiply, divide).**

class Calculator

{

// Method for addition

double add(double a, double b)

{

return a + b;

}

// Method for subtraction

double subtract(double a, double b)

{

return a - b;

}

// Method for multiplication

double multiply(double a, double b)

{

return a \* b;

}

// Method for division

double divide(double a, double b)

{

if (b != 0)

{

return a / b;

}

else

{

print('Cannot divide by zero.');

return double.nan; // Returning NaN for division by zero

}

}

}

void main() {

// Creating an instance of the Calculator class

Calculator calculator = Calculator();

// Performing basic arithmetic operations

double additionResult = calculator.add(5, 3);

double subtractionResult = calculator.subtract(8, 2);

double multiplicationResult = calculator.multiply(4, 6);

double divisionResult = calculator.divide(15, 3);

// Printing the results

print('Addition: $additionResult');

print('Subtraction: $subtractionResult');

print('Multiplication: $multiplicationResult');

print('Division: $divisionResult');

}

**// 12. Implement a class Car with methods to start, stop, and check the fuel level.**

class Car

{

bool isStarted = false;

double fuelLevel = 0.0; // Assuming fuel level is represented as a percentage

// Method to start the car

void start()

{

if (!isStarted)

{

print('Car is starting...');

isStarted = true;

}

else

{

print('Car is already started.');

}

}

// Method to stop the car

void stop() {

if (isStarted) {

print('Car is stopping...');

isStarted = false;

} else {

print('Car is already stopped.');

}

}

// Method to check the fuel level

void checkFuelLevel() {

print('Fuel level: ${fuelLevel.toStringAsFixed(2)}%');

}

// Method to refuel the car

void refuel(double amount) {

if (amount > 0) {

fuelLevel += amount;

if (fuelLevel > 100.0) {

fuelLevel = 100.0; // Cap fuel level at 100%

}

print('Car has been refueled. Fuel level is now ${fuelLevel.toStringAsFixed(2)}%.');

} else {

print('Invalid refuel amount.');

}

}

}

void main() {

// Creating an instance of the Car class

Car myCar = Car();

// Starting the car

myCar.start();

// Checking the fuel level

myCar.checkFuelLevel();

// Refueling the car

myCar.refuel(20.0);

// Checking the fuel level again

myCar.checkFuelLevel();

// Stopping the car

myCar.stop();

}

**// 13. Write a Dart program that uses a function to find the maximum element in a list of integers.**

int findMax(List<int> numbers) {

if (numbers.isEmpty) {

throw ArgumentError('List must not be empty');

}

int max = numbers[0];

for (int number in numbers) {

if (number > max) {

max = number;

}

}

return max;

}

void main() {

List<int> myNumbers = [5, 3, 8, 1, 7, 2, 4, 6];

try

{

int maxValue = findMax(myNumbers);

print('Maximum value in the list: $maxValue');

} catch (e) {

print('Error: $e');

}

}

**//14. Create a class Employee with attributes name, salary, and a method to give a bonus.**

class Employee

{

String name;

double salary;

// Constructor

Employee(this.name, this.salary);

// Method to give a bonus

void giveBonus(double bonusAmount)

{

if (bonusAmount > 0)

{

salary += bonusAmount;

print('$name received a bonus of \$${bonusAmount.toStringAsFixed(2)}. New salary: \$${salary.toStringAsFixed(2)}');

} else {

print('Invalid bonus amount.');

}

}

}

void main()

{

// Creating an instance of the Employee class

Employee david = Employee('Nzeamalu David C.', 50000.0);

// Displaying initial salary

print('${david.name}\'s initial salary: \$${david.salary.toStringAsFixed(2)}');

// Giving a bonus

david.giveBonus(2000.0);

// Displaying updated salary after the bonus

print('${david.name}\'s updated salary: \$${david.salary.toStringAsFixed(2)}');

}

**// 15. Combine a function and a class to calculate the area of a triangle given its base and height.**

class Triangle {

double base;

double height;

// Constructor

Triangle(this.base, this.height);

// Method to calculate the area of the triangle

double calculateArea() {

return 0.5 \* base \* height;

}

}

// Function to calculate the area of a triangle using a class instance

double calculateTriangleArea(double base, double height) {

// Creating an instance of the Triangle class

Triangle myTriangle = Triangle(base, height);

// Calculating and returning the area of the triangle

return myTriangle.calculateArea();

}

void main()

{

// Specifying base and height values

double triangleBase = 5.0;

double triangleHeight = 8.0;

// Calling the function to calculate the area of the triangle

double area = calculateTriangleArea(triangleBase, triangleHeight);

// Printing the area of the triangle

print('The area of the triangle is: $area square units');

}

**// 16. Write a Dart program with a function to find the square root of a given number.**

import 'dart:math';

// Function to find the square root

double findSquareRoot(double number) {

if (number < 0) {

// Handling negative numbers

print('Cannot find the square root of a negative number.');

return double.nan; // Returning NaN for invalid input

}

return sqrt(number);

}

void main() {

// Test cases

double number1 = 25.0;

// Finding and printing the square root of number1

double sqrt1 = findSquareRoot(number1);

print('The square root of $number1 is: $sqrt1');

}

**// 17. Create a function that generates a list of Fibonacci numbers up to a specified limit.**

// Function to generate Fibonacci numbers up to a specified limit

List<int> generateFibonacci(int limit) {

List<int> fibonacciList = [];

int a = 0, b = 1;

while (a <= limit) {

fibonacciList.add(a);

int next = a + b;

a = b;

b = next;

}

return fibonacciList;

}

void main() {

// Specify the limit for Fibonacci numbers

int limit = 100;

// Generate and print Fibonacci numbers up to the specified limit

List<int> fibonacciNumbers = generateFibonacci(limit);

print('Fibonacci numbers up to $limit:');

print(fibonacciNumbers);

}

**// 18. Implement a function to determine if a given year is a leap year.**

// Function to check if a year is a leap year

bool isLeapYear(int year) {

// Leap years are divisible by 4, but not divisible by 100 unless they are also divisible by 400

return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);

}

void main() {

// Test the function with some years

int year1 = 2020;

int year2 = 2021;

int year3 = 2022;

int year4 = 2023;

int year5 = 2024;

int year6 = 2025;

// Check if each year is a leap year and print the result

print('$year1 is${isLeapYear(year1) ? '' : ' not'} a leap year.');

print('$year2 is${isLeapYear(year2) ? '' : ' not'} a leap year.');

print('$year3 is${isLeapYear(year3) ? '' : ' not'} a leap year.');

print('$year4 is${isLeapYear(year4) ? '' : ' not'} a leap year.');

print('$year5 is${isLeapYear(year5) ? '' : ' not'} a leap year.');

print('$year6 is${isLeapYear(year6) ? '' : ' not'} a leap year.');

}

**// 19. Define a recursive function to compute the factorial of a number.**

// Function to compute the factorial of a number

int factorial(int n)

{

// Base case: factorial of 0 or 1 is 1

if (n == 0 || n == 1) {

return 1;

} else {

// Recursive case: n! = n \* (n-1)!

return n \* factorial(n - 1);

}

}

void main() {

// Here is the factorial function with some numbers to see how it works

int number1 = 5;

int number2 = 0;

int number3 = 10;

// Calculate and print the factorial of each number

print('Factorial of $number1: ${factorial(number1)}');

print('Factorial of $number2: ${factorial(number2)}');

print('Factorial of $number3: ${factorial(number3)}');

}

**20. Write a function that takes a list of integers and returns a new list with only the even numbers.**

// Function to filter even numbers from a list

List<int> filterEvenNumbers(List<int> numbers) {

return numbers.where((number) => number % 2 == 0).toList();

}

void main()

{

// Test the filterEvenNumbers function with a list of integers

List<int> inputNumbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20];

// Get the list of even numbers

List<int> evenNumbers = filterEvenNumbers(inputNumbers);

// Print the original and filtered lists

print('Original list: $inputNumbers');

print('List of even numbers: $evenNumbers');

}