Name: Kachhadiya janvi kachhadiya
Roll No: IT25A15
Enroll No: 2025008190013

Question 1:

```
void main(){
 List<int> results = [];

 for(int i=2000; i<=3200;i++){
   if(i%7 == 0 && i%5 != 0){
     results.add(i);
   }
 }

 print(results.join(', '));
}
```

Question 2:

```
import 'dart:io';

void main() {
  print("Enter a number:");
  int number = int.parse(stdin.readLineSync()!);

  bool isPrime = true;

  if (number <= 1) {
    isPrime = false;
  } else {
    for (int i = 2; i <= number ~/ 2; i++) {
      if (number % i == 0) {
        isPrime = false;
        break;
      }
    }
  }

  if (isPrime) {
```

```
    print("$number is a Prime Number");
  } else {
    print("$number is NOT a Prime Number");
  }
}
```

Question 3:

```
import 'dart:io';

void main() {
  print("Enter a sentence:");
  String input = stdin.readLineSync()!;

  int letters = 0;
  int digits = 0;

  for (int i = 0; i < input.length; i++) {
    String ch = input[i];

    if ((ch.codeUnitAt(0) >= 65 && ch.codeUnitAt(0) <= 90) ||
        (ch.codeUnitAt(0) >= 97 && ch.codeUnitAt(0) <= 122)) {
      letters++;
    } else if (ch.codeUnitAt(0) >= 48 && ch.codeUnitAt(0) <= 57) {
      digits++;
    }
  }

  print("LETTERS $letters");
  print("DIGITS $digits");
}
```

Question 4:

```
import 'dart:io';

void main() {
```

```
  print("Enter start range:");
  int start = int.parse(stdin.readLineSync()!);

  print("Enter end range:");
  int end = int.parse(stdin.readLineSync()!);

  for (int i = start; i <= end; i++) {
    if (i % 2 == 0) {
      print("Number: $i  Square: ${i * i}");
    }
  }
}
```

Question 5:

```
import 'dart:io';

class AgeException implements Exception {
  String message;
  AgeException(this.message);

  @override
  String toString() => message;
}

void main() {
  try {
    print("Enter your age:");
    int age = int.parse(stdin.readLineSync()!);

    if (age < 18) {
      throw AgeException("Age must be 18 or above.");
    }

    print("Access granted. Age is valid.");
  } catch (e) {
    print("Exception: $e");
  }
```

```
}
```

Question 6:

```
import 'dart:io';

class Task {
  String name;
  bool isCompleted;

  Task(this.name, {this.isCompleted = false});
}

void main() {
  List<Task> tasks = [];
  bool running = true;

  while (running) {
    print("\n--- TO-DO LIST MENU ---");
    print("1. Add Task");
    print("2. Remove Task");
    print("3. Mark Task as Completed");
    print("4. View Pending Tasks");
    print("5. View Completed Tasks");
    print("6. Exit");
    print("Enter your choice:");

    int choice = int.parse(stdin.readLineSync()!);

    switch (choice) {
      case 1:
        print("Enter task name:");
        String taskName = stdin.readLineSync()!;
        tasks.add(Task(taskName));
        print("Task added successfully.");
        break;

      case 2:
        displayAllTasks(tasks);
```

```
        print("Enter task number to remove:");
        int index = int.parse(stdin.readLineSync()!) - 1;
        if (index >= 0 && index < tasks.length) {
          tasks.removeAt(index);
          print("Task removed.");
        } else {
          print("Invalid task number.");
        }
        break;

      case 3:
        displayAllTasks(tasks);
        print("Enter task number to mark as completed:");
        int index = int.parse(stdin.readLineSync()!) - 1;
        if (index >= 0 && index < tasks.length) {
          tasks[index].isCompleted = true;
          print("Task marked as completed.");
        } else {
          print("Invalid task number.");
        }
        break;

      case 4:
        print("\n--- Pending Tasks ---");
        for (var task in tasks) {
          if (!task.isCompleted) {
            print("- ${task.name}");
          }
        }
        break;

      case 5:
        print("\n--- Completed Tasks ---");
        for (var task in tasks) {
          if (task.isCompleted) {
            print("- ${task.name}");
          }
        }
        break;
```

```
      case 6:
        running = false;
        print("Exiting To-Do List...");
        break;

      default:
        print("Invalid choice. Try again.");
    }
  }
}

// Helper function to display all tasks
void displayAllTasks(List<Task> tasks) {
  print("\n--- All Tasks ---");
  for (int i = 0; i < tasks.length; i++) {
    print("${i + 1}. ${tasks[i].name} "
        "[${tasks[i].isCompleted ? 'Completed' : 'Pending'}]");
  }
}
```

Question 7:

```
import 'dart:io';

class BankAccount {
  double _balance = 0.0;

  void deposit(double amount) {
    if (amount > 0) {
      _balance += amount;
      print("Deposit successful. Amount: ₹$amount");
    } else {
      print("Invalid deposit amount.");
    }
  }

  void withdraw(double amount) {
    if (amount <= 0) {
```

```
      print("Invalid withdrawal amount.");
    } else if (amount > _balance) {
      print("Insufficient balance.");
    } else {
      _balance -= amount;
      print("Withdrawal successful. Amount: ₹$amount");
    }
  }

  void checkBalance() {
    print("Current Balance: ₹$_balance");
  }
}

void main() {
  BankAccount account = BankAccount();

  account.deposit(5000);
  account.withdraw(2000);
  account.checkBalance();
  account.withdraw(4000);
}
```

Question 8:

```
// Base class
class Employee {
  String name;
  double basicSalary;

  Employee(this.name, this.basicSalary);

  double calculateSalary() {
    return basicSalary;
  }
}

class PermanentEmployee extends Employee {
```

```
  PermanentEmployee(String name, double basicSalary)
      : super(name, basicSalary);

  @override
  double calculateSalary() {
    double hra = basicSalary * 0.20;
    double da = basicSalary * 0.10;
    return basicSalary + hra + da;
  }
}

class ContractEmployee extends Employee {
  int hoursWorked;
  double hourlyRate;

  ContractEmployee(String name, this.hoursWorked, this.hourlyRate)
      : super(name, 0);

  @override
  double calculateSalary() {
    return hoursWorked * hourlyRate;
  }
}

void main() {
  Employee emp1 = PermanentEmployee("Amit", 30000);
  Employee emp2 = ContractEmployee("Ravi", 120, 500);

  print("Permanent Employee Salary: ₹${emp1.calculateSalary()}");
  print("Contract Employee Salary: ₹${emp2.calculateSalary()}");
}
```

Code 1:

```
// 1. Given Code
// class Student {
// int id;
// String name;
// Student(int id, String name) {
// id = id;
// name = name;
// }
// }
// void main() {
// Student s = Student(1, "Rahul");
// print(s.name);
// }
// Tasks -
// • Does the code produce output or error?
// • If error → identify and rectify it
// • Add functionality to:
// ○ Display both id and name
// ○ Add a method display()

// Resolve code
class Student {
    int? id;
    String? name;
    Student(int id, String name) {
        this.id = id;
        this.name = name;
    }

    //add Display method
    void display() {
        print("ID: $id");
        print("Name: $name");
      }
}
void main() {
    Student s = Student(1, "Rahul");
    s.display();
```

```
}
```

Code 2:

```
// 2. Given Code
// void main() {
// int marks = 85;
// if (marks > 90) {
// print("A");
// } else if (marks > 75) {
// print("B");
// } else {
// print("C");
// }
// }

// Tasks-
// • Predict output
// B

// • Identify logical issue
// | Marks    | Grade |
// | -------- | ----- |
// | ≥ 90     | A     |
// | 75 - 89  | B     |
// | < 75     | C     |

// • Fix grading logic
import 'dart:io';

void main() {
  print("Enter marks:");
  int marks = int.parse(stdin.readLineSync()!);

  if (marks >= 90 && marks <= 100) {
    print("Grade: A");
  } else if (marks >= 75 && marks < 90) {
    print("Grade: B");
```

```
  } else if (marks >= 0 && marks < 75) {
    print("Grade: C");
  } else {
    print("Invalid marks");
  }
}
```

Code 3:

```
// 3. Given Code -
// class Product {
// String name;
// double price;
// Product(this.name, this.price);
// }
// void main() {
// Product p = Product("Laptop", -50000);
// print(p.price);
// }
// Tasks -
// • Is this logically correct?
// • Add validation using exception handling
// • Add functionality: Apply 10% discount if price > 30000

class Product {
  String name;
  double price;

  Product(this.name, this.price) {
    if (price <= 0) {
      throw Exception("Invalid price: Price must be greater than 0");
    }
  }

  void applyDiscount() {
    if (price > 30000) {
      price = price - (price * 0.10);
    }
  }
}
```

```
void main() {
  try {
    Product p = Product("Laptop", 50000);
    p.applyDiscount();
    print("Final Price: ₹${p.price}");
  } catch (e) {
    print("Error: $e");
  }
}
```

Code 4:

```
// 4. Given Code -
// class Person {
// String name;
// Person(this.name);
// }
// void main() {
// Person p1 = Person("Amit");
// Person p2 = Person("Amit");
// print(p1 == p2);
// }

// Tasks-
// • Predict output
// false

// • Explain why
// In Dart, == compares object references by default
// p1 and p2 are two different objects in memory
// Even though their name values are the same, references are different

// • Override equality operator
// To compare objects by value, we override:
// == operator
// hashCode

// • Add functionality: compare by name
class Person {
```

```
  String name;

  Person(this.name);

  @override
  bool operator ==(Object other) {
    if (identical(this, other)) return true;
    if (other is! Person) return false;
    return name == other.name;
  }

  @override
  int get hashCode => name.hashCode;
}

void main() {
  Person p1 = Person("Amit");
  Person p2 = Person("Amit");
  Person p3 = Person("Rahul");

  print(p1 == p2); // true
  print(p1 == p3); // false
}
```

Code 5:

```
// 5. Given Code -
// void main() {
// int day = 1;
// switch (day) {
// case 1:
// print("Monday");
// case 2:
// print("Tuesday");
// break;
// }
// }
// Tasks -
// • Predict output
// Monday
```

```dart
// Tuesday

// • Explain Dart switch behavior
// Dart allows fall-through if break is not used
// Since case 1 has no break, execution continues to case 2
// This is called fall-through

// • Fix the logic
void main() {
  int day = 1;

  switch (day) {
    case 1:
      print("Monday");
      break;
    case 2:
      print("Tuesday");
      break;
  }
}

// • Add functionality: handle all weekdays
void main() {
  int day = 3;

  switch (day) {
    case 1:
      print("Monday");
      break;
    case 2:
      print("Tuesday");
      break;
    case 3:
      print("Wednesday");
      break;
    case 4:
      print("Thursday");
      break;
    case 5:
      print("Friday");
```

```
      break;
    case 6:
      print("Saturday");
      break;
    case 7:
      print("Sunday");
      break;
    default:
      print("Invalid day");
  }
}
```