Logical Programs

1. **Fibonacci Series**

   Fibonacci series is a special type of series in which the next term is the sum of the previous two terms. For example, if 0 and 1 are the two previous terms in a series, then the next term will be 1(0+1).

2. **Perfect Number**

   a. Just like the Armstrong number, the Perfect Number is also a special type of positive number. When the number is equal to the sum of its positive divisors excluding the number, it is called a Perfect Number. For example, 28 is the perfect number because when we sum the divisors of 28, it will result in the same number. The divisors of 28 are 1, 2, 4, 7, and 14. So,

   b. 28 = 1+2+4+7

   c. 28 = 28

3. **Prime Number**

   Just like the Perfect number, the Prime number is also a special type of number. When the number is divided greater than 1 and divided by 1 or itself is referred to as the Prime number. 0 and 1 are not counted as prime numbers. All the even numbers can be divided by 2, so 2 is the only even prime minister.

4. **Reverse a number**

   In Java, we can reverse a number either by using for loop, while loop, or using recursion. The simplest way to reverse a number is by using for loop or while loop. In order to reverse a number, we have to follow the following steps:

   a. We need to calculate the remainder of the number using the modulo

   b. After that, we need to multiply the variable reverse by 10 and add the remainder into it.

   c. We then divide the number by 10 and repeat steps until the number becomes 0.

5. **Coupon Numbers**
    a. Desc -> Given N distinct Coupon Numbers, how many random numbers do you need to generate a distinct coupon number? This program simulates this random process.
    b. I/P -> N Distinct Coupon Number
    c. Logic -> repeatedly choose a random number and check whether it's a new one.
    d. O/P -> total random number needed to have all distinct numbers.
    e. Functions => Write Class Static Functions to generate random numbers and to process distinct coupons.
6. **Simulate Stopwatch Program**
    a. Desc -> Write a Stopwatch Program for measuring the time that elapses between the start and end clicks
    b. I/P -> Start the Stopwatch and End the Stopwatch
    c. Logic -> Measure the elapsed time between start and end
    d. O/P -> Print the elapsed time.

Programs for JUnit Testing

1. **Find the Fewest Notes to be returned for Vending Machine**
    a. Desc -> There is 1, 2, 5, 10, 50, 100, 500 and 1000 Rs Notes which can be returned by Vending Machine. Write a Program to calculate the minimum number of Notes as well as the Notes to be returned by the Vending Machine as a Change
    b. I/P -> read the Change in Rs to be returned by the Vending Machine
    c. Logic -> Use Recursion and check for largest value of the Note to return change to get to the minimum number of Notes.
    d. O/P -> Two Outputs - one the number of minimum Note needed to give the change and second list of Rs Notes that would given in the Change
2. To the Util Class add ***dayOfWeek*** static function that takes a date as input and prints the day of the week that date falls on. Your program should take three command-line arguments: m (month), d (day), and y (year). For m use 1 for January, 2 for February, and so forth. For output print 0 for Sunday, 1 for Monday, 2 for Tuesday, and so forth. Use the following formulas, for the Gregorian calendar (where / denotes integer division):

    $y_0 = y - (14 - m) / 12$

    $x = y_0 + y_0/4 - y_0/100 + y_0/400$

    $m_0 = m + 12 \times ((14 - m) / 12) - 2$

    $d_0 = (d + x + 31m_0 / 12) \bmod 7$
3. To the Util Class add ***temperaturConversion*** static function, given the temperature in fahrenheit as input outputs the temperature in Celsius or viceversa using the formula

    Celsius to Fahrenheit: (°C × 9/5) + 32 = °F

    Fahrenheit to Celsius: (°F − 32) x 5/9 = °C
4. Write a Util Static Function to calculate ***monthlyPayment*** that reads in three command-line arguments P, Y, and R and calculates the monthly payments you would have to make over Y years to pay off a P principal loan amount at R per cent interest compounded monthly. The formula is The formula is

```
                 P r
payment =  ---------------,  where n = 12 * Y, r = R / (12 * 100)
           1  - (1 + r)^(-n)
```

5. Write a static function ***sqrt*** to compute the square root of a nonnegative number c given in the input using Newton's method:

   - initialize t = c

   - replace t with the average of c/t and t

   - repeat until desired accuracy reached using condition Math.abs(t - c/t) > epsilon*t where epsilon = 1e-15;

6. Write a static function ***toBinary*** that outputs the binary (base 2) representation of the decimal number typed as the input. It is based on decomposing the number into a sum of powers of 2. For example, the binary representation of 106 is 11010102, which is the same as saying that 106 = 64 + 32 + 8 + 2. Ensure necessary padding to represent 4 Byte String.

   To compute the binary representation of n, we consider the powers of 2 less than or equal to n in decreasing order to determine which belong in the binary decomposition (and therefore correspond to a 1 bit in the binary representation).

7. Write Binary.java to read an integer as an Input, convert to Binary using toBinary function and perform the following functions.

   i. Swap nibbles and find the new number.

   ii. Find the resultant number is the number is a power of 2.

   A nibble is a four-bit aggregation, or half an octet. There are two nibbles in a byte.

   Given a byte, swap the two nibbles in it. For example 100 is to be represented as 01100100 in a byte (or 8 bits). The two nibbles are (0110) and (0100). If we swap the two nibbles, we get 01000110 which is 70 in decimal.