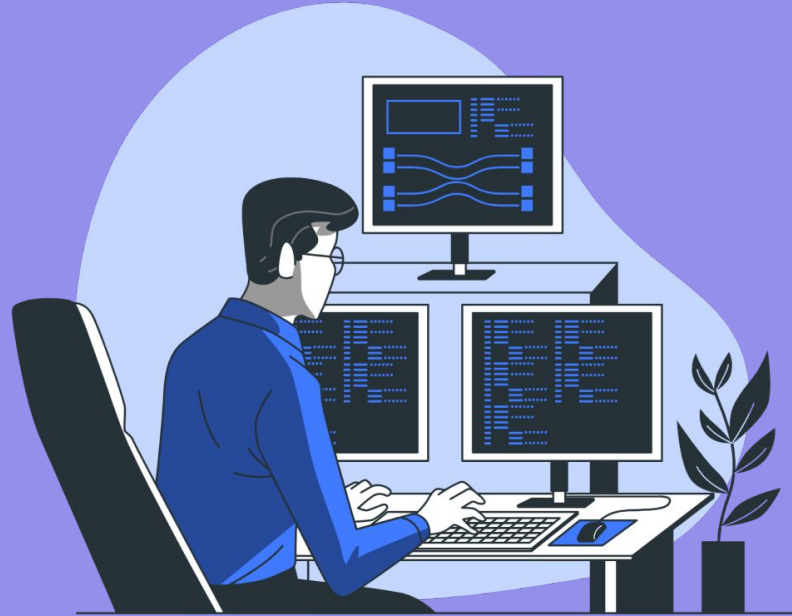# Advanced Problem Solving: Conditional Statements, Iterative Statements, Functions

Relevel
by Unacademy

1. Height Of Prism (15 min):

Problem Statement:

Given the volume and perimeter of the triangular base of the prism, and the circumference of the inner circle. Calculate the height of the prism.

Example:

Input	radius=3, perimeter=15, Volume=270
Output	12

Explanation:

The radius of the triangle is given as `area / semi perimeter`. And the semi perimeter of the triangle is half the triangle perimeter. Substituting the values we can get the area of the triangle i.e.

area = Inradius * semi perimeter

= 3 * 15 /2

Comparing it with volume of prism i.e. area * h we can get the height as

270 = h * (3 * 15 / 2)
h = 12

<u>Solution:</u>

```
let radius = 3;
let perimeter = 15;
let volume = 270;

let semiperimeter = perimeter / 2;
let area = radius * semiperimeter;
let height = volume / area;
console.log("` of the prism is
",height," cm");
```

LightWeight Solution:

```javascript
let radius = 3;
let perimeter = 15;
let volume = 270;

let height = volume / (radius *
(perimeter / 2));
console.log("Height of the prism
is ",height," cm");
```

2. Distance to Office (15 min):

Problem Statement:

A man reaches his office `y` min late when he travels at x km/hr. If he travels at x+2 km/hr then he reaches the office `y-10` min early. Find the distance between his home and office given the value of x and y.

Example:

Input	x=4, y=20
Output	6

Example Explanation:

Suppose the distance between the office and home is D and the time he takes to reach the office is T minutes.

Case-1: when he reaches the office 20 minutes late the time taken to cover the distance D would be T+20

Case-2: When he reaches the office 20-10 i.e. 10 minutes earlier then the time taken taken to cover D is T-10

Now in both the cases Distance is same, therefore `(T+20)*4` = (T-10)*6 . On solving we get T = 70 minutes.

Now using this T in any of the case say case 1 we get D = 90 * 4 = 360 minutes * km / hr = 6 km/hr

Relevel
by Unacademy

```
var x = 4;
var y = 20;
let time = y * (x+1) - 5 * (x+2) ;
let distance = ( (y+time) / 60 ) * x ;
console.log("Distance between the house
and office is : ",distance," km");
```

Solution Explanation:

Suppose the Distance between the office and home is D and the time it takes to reach the office is T minutes.

Case-1: when he reaches the office 20 minutes late, the time taken to cover the distance D would be T+y

Case-2: When he reaches the office 20-10 i.e. 10 minutes earlier than the time taken to cover D is T-(y-10)

Now in both cases, Distance is the same, therefore

(T+y) * x = (T-(y-10)) * (x + 2)

On simplifying

T = y(x+1) - 5(x-2)

Total time to cover distance D is (y+time) minutes, i.e. (y+time)/60 hours

Speed to cover distance is x

Using Distance = Speed x Time = x * (y+time)/60

3. Bob the Builder (15 min):

Problem Statement:

Bob has recently started his career as a builder. But he is always confused about the number of bricks that will be needed at the site. To be on safer side he always orders more than the required number of bricks. This inturn helps him in saving the transportation cost of bricks but adds an unnecessary burden on the pockets of the client. Due to which most of the clients never call Bob back for any new project.

Bob needs your help in deciding how many bricks are needed for constructing a wall of length L meter, height H meter and thickness T meter if 15% of the wall is filled with mortar and the dimension of 1 brick is 24x12x8 cm.

Input	L=24, H = 8, T = 0.6
Output	42500

Explanation:

Since 15% of the wall is filled with mortar therefore the volume of the wall that needs to be filled with brick is 85%.

Volume of 1 brick is 24x12x8 cm^3

Therefore the amount of brick needed to fill 85% of the volume of the wall is:

Number of bricks = (0.85 * 24 * 8 * 0.6 * 100^3) / (24 * 12 * 8)

<u>Solution:</u>

```
let length = 24;
let breadth = 8;
let thickness = 0.6;

let brick_volume = 24*12*8;
let mortar_percent = 15;
let wall_volume = length * breadth * thickness;
```

```
let no_bricks = Math.ceil(((100 - mortar_percent) / 100 ) * wall_volume * (
100*100*100 )/ brick_volume) ;

console.log("Number of bricks needed to construct the wall is: ",no_bricks);
```

LightWeight Solution:

```
let length = 24;
let breadth = 8;
let thickness = 0.6;
let no_bricks = Math.ceil( 0.85 * length * breadth * thickness * Math.pow(100,3) /
2304) ;
console.log("Number of bricks needed to construct the wall is: ",no_bricks);
```

4. LCM of number (15 min) :

Problem Statement:

Given two numbers, find the least possible positive integer that is divided by them.

Example-1:

Input	10,5
Output	10

Example-2:

Input	12,16
Output	80

Explanation-1:

2	10,5
5	5,1
	1,1

L.C.M = 2 * 5 = 10

Explanation-2:

2	12,16
2	6,8
2	3,4
2	3,2
3	3,1
	1,1

L.C.M = 2 * 2 * 2 * 2 *5 = 80

Solution :

```
function LCM(num1, num2){
    if(num1 > num2)
        greater = num1;
    else
        greater = num2;

    while(true){
        if((greater % num1 == 0) && (greater % num2 == 0)){
            lcm = greater;
            break;
        }
        greater += 1;
        }
    return lcm
    }
```

```
num1 = 10;
num2 = 5;

console.log("L.C.M. :", LCM(num1, num2))
```

Explanation:

While taking out the LCM of given two numbers, say A, B, it is obvious that the number divisible by both the numbers will be greater or equal to the max of A, B.
Now we can iterate from max(A,B) and check if that number is divisible by A and B both or not. If yes, that number will be our LCM.

Note to Educator: An optimized solution for this would be to use the knowledge that the product of two numbers is equal to the product of gcd and lcm. Students are supposed to work on it as homework; therefore, it is given as a homework assignment.

LightWeight Solution

```javascript
const findGCD = (num1, num2) => {
    let a = Math.abs(num1);
    let b = Math.abs(num2);
    while (a && b && a !== b) {
        if(a > b){
            [a, b] = [a - b, b];
        }else{
            [a, b] = [a, b - a];
        };
    };
    return a || b;
};



num1 = 10;
num2 = 5;

console.log("L.C.M. :", (num1*num2) /findGCD(num1, num2))
```

Explanation:

LCM of the number can also be calculate by the formula

LCM = (num1*num2) / GCD

where GCD is the greatest common divisor i.e. the highest common factor of both the numbers.

5. Sum of Digits (15 min) :

Problem Statement:
Given a positive number you need to write a program in javascript to find the sum of the inner digits of that number.

Example-1:

Input	2124
Output	3

Example-2:

Input	70
Output	70

Explanation - 1:

The Inner digits of the number 2124 are 1 and 2 whose sum is 3

Explanation - 2:

The Inner digits of the number 70 is None therefore the output will be -1.

Note: Same is the case for number with single digit

<u>Solution:</u>

```
function getDigitSum(num)
{
    var sum = 0;
    if (num.toString().length<3){
        return -1;
    }
    do{
     num = parseInt(num / 10);
      sum = sum + num % 10;
```

```
        }
    while (parseInt(num / 100) != 0);
    return sum;
}


var n = var n = 2124;
console.log(getDigitSum(n));
```

Solution Explanation:

Since the internal digit can happen only on more than two digits, we first need to check that. If the condition fails, we simply return -1.
Now we need to perform the addition of the middle elements. To skip the rightmost element, we are first dividing it by 10.
So say if the number is 1234, then after dividing by ten, we are left with 123
i.e., on dividing by 10, we get rid of the leftmost digit of the number.
Therefore in *iteration-1*
num =1234 / 10 = 123

sum = 0 + 123 % 10 = 0+3 = 3

Since 123%100 = 3 we move to iteration 2

Iteration-2

num=123/10 = 12

Sum = 3 + 12%10 = 3+2 = 5

Now 12 %100 is 0 therefore we break the loop and return the sum. Which in this case is 5.

6. Power of a Number (20 min) :

Problem Statement:
Given two numbers A and B, You are supposed to find A to power B without using any built -in function.

Example:

Input	num=10, power=2
Output	100

Explanation:

10 ^ 2 = 100

Solution:

```javascript
function computePower(num, power)
{
    let result = 1;

    while (power > 0) {
        result = result * num;
    power -=1;
    }
    return result;
}
let number = 10;
let power = 2;
console.log(computePower(number,power));
```

Solution Explanation:

To compute the power of the number without using the in built function, we can simply multiply the number to itself the given number of times. I.e. $4 \wedge 3 = 4 * 4 * 4 = 64$.

Optimized Solution

```javascript
function computePower(num, power)
{
    let result = 1;

    while (power > 0) {
    if (power & 1)
            result = result * num;

        power = power >> 1;
        num = num * num;     }
    return result;
}
let number = 10;
let power = 2;

console.log(computePower(number,power));
```

Solution Explanation:

In the previous solution the while loop was iterating n number of time, where n was equal to the value of power. But in the optimized solution this iteration is reduced to half by using a right shift operator for each iteration. Whenever the power value becomes odd, we can multiply the number with the resultant and store it in the resultant else; we simply update the number by multiplying it.
To check if number is even or odd We can perform a bit-wise operation. If the result is one, then the number is odd else even.

7. Summation Less Adding (15 min)

Problem Statement:

Given two numbers A and B, You are supposed to find the sum of A and B without using the addition operator i.e. `+`

Example:

Input	A = 10 B = 5
Output	15

Explanation:

10 + 5 = 15

Solution:

```
function add(num1, num2)
{
    while(num2!=0){
            let carry = num1 & num2;
        num1 = num1 ^ num2 ;
        num2 = carry << 1 ;
    }
    return num1;
}
let num1 = 10;
let num2 = 5;

console.log(num1 ," + ",num2," = ",add(num1,num2));
```

Explanation:

Bitwise and `&` operators return 0 if either of the bits are 0.
Bitwise XOR `^` operator return 1 if both the bits are different

Knowing these two, we can add two numbers on their binary by keeping the count of sum and carry. Where sum
was performed using the bitwise xor operator, and carry was performed using bitwise.

Here Num1 will be used to store the sum result while num2 will be used as the counter to terminate the loop

Dry Run:
Suppose we need to do 10 + 5 then in there binary form it will be
10 = 1010
5 =   101
Till the time 101 !=0 get
carry = 1010 & 101 = 0000

Sum = 1010 ^ 101 = 1111
Shiting carry to right by 1 , carry remains 0000
Since 0000 therefore we return 1111 which is 15

8. Nth Fibonacci (15 min)

Problem Statement:
Given the value of n, find out the nth term of the fibonacci series.
Fibonacci series is like 0,1,1,2,3,5,8,13... where the next element is always the sum of the previous two elements and the first two elements are 0 and 1.

Example-1:

Input	5
Output	3

Example-2:

Input	24
Output	28657

Explanation - 1:

Given the Fibonnaci series as 0,1,1,2,3,5,8,13.. the 5th term is 3

Explanation - 2:

Given the Fibonnaci series as 0,1,1,2,3,5,8,13.. the 5th term is 28657

Solution

```
function nFibb(Nth){
  let prev = 0;
  let curr = 1;
  let temp;

  if(Nth == 1)
    return prev;
  else if(Nth==2)
    return curr;
  else
    for (let idx=3;idx<=Nth;idx++){
      temp = prev;
      prev=curr;
      curr+=temp;
```

```
      }
   return curr;
}



let Nth = 8
console.log(Nth,"th Fibonacci number is",nFibb(Nth));
```

Explanation:

If the nth term is 1 then return 0

If the nth term is 2 then return 1

In other cases, iterate from 2 to N abd take the current and previous element and update the current element. Finally, return the current term.

9. 60 a Factor (20 min)

Problem Statement:

Given a number A you are supposed to take the digits of that number, and find a permutation such that 60 becomes a factor of that permutation. If there exists such permutation then return 1 else 0.

Input	2340
Output	1

Example-2:

Input	24
Output	0

Explanation-1:
Given 2340, a possible permutation of its digit which is divisible by 60 can be 4320. This is equivalent to 60*72 therefore the result is 1.
 Explanation-2:
Given 24, There isn't any permutation of the digit whose factor is 1, as there permutations are 42, 24. therefore the result is 0.

Solution:

```
function isDivisibleBy60( num ){

    if(num%10==0)
        return 1;

    sum = 0;
    last = false;
    secondLast = false;
    while(num!=0){
        i = num%10;
        num=parseInt(num/10);
        if(i%2==0 && (i>0 || last))
            secondLast = true;
        if(i==0)
            last = true;
```

```
        if(sum%3==0 && last && secondLast)
            return 1;
        return 0;
        }
let number = 2340;
console.log(isDivisibleBy60(number));
```

Explanation:

60 is a composite number and any number which is divisible by composite numbers is supposed to be divisible by the prime number raised to the highest power. Which in case of 60 are 4*5*3.
So any number that is divisible by 3,4,5 will be divisible by 60.
  1.   Now for number to be divisible by 5, last digit is supposed to be 0 or 5
  2.   Number is divisible by 4 if its last two digits are divisible by 4.
On combining the above two points, we can conclude that any number which needs to be divisible by both 4,5 needs to have 0 as the last digit and that second last digit as even.

Finally to check if the number is divisible by 3, we need to sum the digits and see if it goes by 3.
Using all these 3 we can come up with the solution to figure out if there exists a digit permutation which is divisible by 60.

10. Sum OF Prime (20 min):

Problem Statement:
Given a number A which is an even integer, you need to print two prime numbers whose sum will be equal to A.
Constraint: A>2

Example-1:

Input	4
Output	2,2

Example-2:

Input	15
Output	2,13

Explanation-1:

2 is prime number, therefore 2+2 is 4

Explanation-2:

2 and 13 are prime numbers whose sum is 15.

Note: There Can be multiple solutions to this problem for a given number. Since this is a practice session, you can print any one of the correct answers.

```
function isPrime(n)
{
    for(let i=2;i<=Math.sqrt(n);i++)
    {
        if(n%i==0) return false;
    }
    return true;
}

function sumPrime(n) {
    for(let i=2;i<=n/2;i++)
    {
        if( isPrime(i) && isPrime(n-i) )
        {
            console.log(i, n-i);
            return;
        }
    }
```

```
}
let num = 15;
sumPrime(num);
```

For the range of values from 2 to A, start iterating the numbers from both the ends. Example if A is 8 then the range will be 2, 3, 4, 5, 6, 7
Since we need to find pair, Select the number `i` and `A-i`.
Check if both are prime or not. If both are prime then log the output and return.
We will iterate in the range till the middle range so as to keep position `i` <= `A-i` as comparison above. That will just be a repetitive task.
Lets understand this by taking the above range as example:

Range = 2,3,4,5,6,7
A= 8
Let the pair be p1 and p2
Iteration-1
p1 = 2   & p2 = 6

2 is prime but 6 is not therefore continue

Iteration-2
p1 = 3   & p2 = 5
Both 3 and 5 are prime, hence log there output and return.

And to check prime, we are simply trying to find a factor of the number starting from 2 to square root of it. If there exists a factor, then the number is not prime.

MCQ's:

1. Output of the following script:

```
let result = 0;
for (let i = 0; i < 5; i++) {
  result += i;
}
console.log(result);
```

      a) 10
      b) 0
      c) 5
      d) None of the above

Answer : a) 10

2. Output of the following script:

```
let num=16;
console.log(num<<3);
```

a) 16
b) 3
c) 0
d) 128

Answer d) 128

3. Which of the operators gives 1 when neither of the digits are the same.

a) OR
b) XOR
c) AND
d) NOT

Answer b) XOR

4. 6 th element of the fibonacci series is?

a) 5
b) 4
c) 3
d) 2

Answer a) 5

5. How to get the last 3 digits of number A.

a) A%10
b) A%100
c) A%1000
d) A%10000

Answer c) A%1000

THANK YOU