TP Session 4

30 September 2024 16:27

Given x number of 1's followed by y number of 0's, your task is to find the decimal representation of the x and y.

```
    int decimal = Math.pow(2, x) - 1
     decimal =decima1 * Math.pow(2, y)
```

Better approach

```
         int res=0
         int decimal = (1 << x) - 1
         decimal <<= y
```

if number is too large use big integer class

```
        import java.math.BigInteger;
         BigInteger x = scanner.nextBigInteger();
         BigInteger y = scanner.nextBigInteger();

          BigInteger res = BigInteger.ZERO;

         BigInteger res =
BigInteger.ONE.shiftLeft(x.intValue()).subtract(BigInteger.ONE).shiftLeft(y.intValue());
```

Given Xth and Yth Bit position. Create a number where X and Yth bit are Set:

```
int result= (1<<x) | (1<<y)
```

In a given integer - N, check whether the ith bit is set or not.

suppose i=2 and N=10, 1010 in binary: 2nd bit is 0(starting from 0,1,2)

```
if(n>>i & 1==1) then
    print true
else
    print false
```

Calculate number of bits required to represent an integer value:

```
                while n>0 do
                 count++
                 n>>=1
                 print count
```

instead: `int bits = (int)(Math.log10(num) / Math.log10(2)) + 1;`

For example Take num=4

---

this is equal to $\log_{10}(4)/\log_{10}(2)$--> $\log_2(4)=2+1=3$

Reverse the Bits of an integer number and print the value in decimal.

```
    while N>0 Do
            res=0
           res = (res << 1) | (n & 1);
            n >>= 1;
        }
       print res
```

N&1 to extract LSB bit of original number.
Res<<1 to shift the result bit to the left side so that we can reverse the original number.
OR operation to combine the shifted res with the extracted bit.
n>>=1 to move to the next bit in the number

Largest Power of 3 less than or equal to given number N:

```
initalize res to 1
initalize power to 1

until power < N Do
   res=power
   power =power * 3

print(res)
```

If the number is too large use BigInteger Class

```
-----------------------------------------------------
   String input = scanner.nextLine();
   BigInteger N = new BigInteger(input);
         // or read directly as: BigInteger x = scanner.nextBigInteger();

       BigInteger res = BigInteger.ONE;
       BigInteger power = BigInteger.ONE;

           while (power.compareTo(N) < 0) {
           res = power;
           power = power.multiply(BigInteger.valueOf(3));
       }

       print(res)
```

Print Number of trailing zeroes in the factorial of a given number N

GENERAL APPROACH

```
     mainfunction()
     {
        read long int value
        long fact=factorial(n)
        long count=counttrailingzeroes(fact)
        print(count)
```

```
}

factorial(long n)
{
      intitalize long res=1
      for i=1 to n   (i=1;i<=n;i++)
      res=res*i
      return res
}

counttrailingzeroes(long n)
  {
     int count = 0;
    while n > 0 Do
      if n mod 10 == 0 then
         count++
      else
         break the loop
      n= n/10;
    END WHILE
     return count
  }
```

BETTER APPROACH

every multiple of 5 contributes to number of trailing zeroes in N!

use formula: n/5+n/25+n/125 and so on!!

initalize count=0

```
while n >0 Do
count = count + (n/5)
n = n/5
END WHILE

print(count)
```

Check if a given number is Prime Number or Not

```
int flag=1
if n<=1
   print not Prime
   return

for i=2 to Math.sqrt(n) DO
   if n mod i==0 then
   set flag to 0
    break the loop

if flag==1 then
   print "prime"
else
   print "not Prime"
```

Instead of using Math.sqrt the better efficient way could be:

```
for i=2 to i*i<=n
   if n mod i==0
      set flag=0
      break the loop
```

Print Prime numbers from 1 to given range N

```
mainfunction()
{
      read n
      for i=1 to n+1 Do
       if(checkprime(i)):
           System.out.print(i+" ")
}

boolean checkprime(int n)
{
    if n is less than or equal to 1
        return False
    for i=2 to Math.sqrt(n) Do
        if n%i==0
            return false
    return true
}
```

Print Prime Numbers upto given Count

```
read N
set count=0
set i=1
while count<n Do:
    res=checkprime(i)
    if res==true
        System.out.print(i+" ")
        increment count++
    increment i++


boolean checkprime(int n)
{
    if n is less than or equal to 1
        return False
    for i=2 to Math.sqrt(n) Do
        if n%i==0 then
            return false
    return true
}
```

SIEVE OF ERATOSTHENES: THE MOST OPTIMIZED PRIME NUMBER LOGIC

```
        boolean[] prime = new boolean[n + 1]
```

```
        for int i=0 to n Do
          set prime values to true


        OuterLoop for int i=2 to i*i<=n Do
           check if prime[i] is true if so then gotoinner loop
                  for int j=p*p to n (Note: increment innerloop by j=j+p)
                     set prime[j] to false


           for int i=2 to n Do
            if prime[i] is true then
              print(prime[i]+" ")
```

Print prime numbers upto given range N for T test cases each on new line

```
Take input number of test cases T
iterate from i=1 to T Do
    read integer number n
   call printupton(n) function


function printupton(n):
    iterate from i to n Do
        if(checkforrime(i)==true):
            print(i+" ")
    println()
End Function


function boolean checkforprime(int n)
    if n<=1 then
        return false
    iterate from i=2 to sqrt(n) D0
        if n%i==0 then
            return false
    return true
End function
```

for example:

```
T=3
10
2 3 5 7
20
2 3 5 7 11 13 17 19
100
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

Problems

28 September 2024 11:34