Operator Precedence -

tells about order of execution ie how operators will be evaluated in an expression.

eg - int ans $= 2 * \boxed{\frac{3}{4}} + 5$

$= 2 * 0 + 5$

$= 5$

There is always ambiguity as to which operator should be executed first, therefore predence is needed.

Table → () [] -> ++ --	Postfix	↶ Highest precedence
+ - ! ~ ++	Unary	
* / %	Multiplicative	
+ -	Additive	
<< >>	Shift	
< <= > >=	Relational	
== !=	Equality	
&	Bitwise	
^		
|		
&&	Logical AND	
||	" OR	
?:	Conditional	
=, +=, -=, *=, /=		

Associativity - Helps to tell which operand will be executed first if they have same precedence.

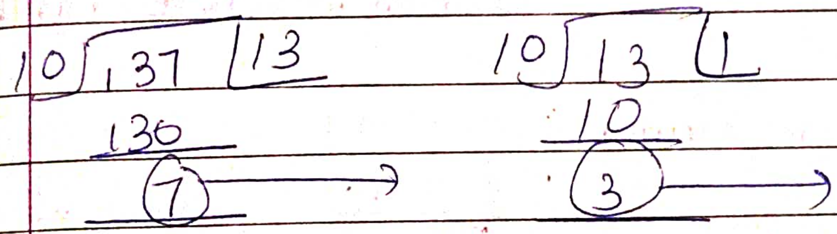
eg $\left[\left(\frac{10}{10}\right) * 10\right]$ * and / → same precedence

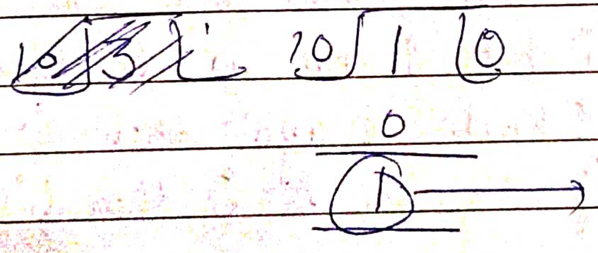
Associativity → Left to right So left operand evaluated first.

$1 * 10$

→ 10.

137 → Seperate all digits





Divide by 10 we get the seperate digit.

input → 137

```
while (n!=0)
{
  int digit = n% 10;
  cout <<digit : "<< digit <<endl;
  n=n/10;
}
cout <<"DONE" <<endl;
```

output

```
digit : 7
digit : 3
digit : 1
```

Digit → No → 1, 3, 7

$$\begin{array}{ccc} 1, & 3, & 7 \\ \downarrow & \downarrow & \downarrow \\ 10^2 & 10^1 & 10^0 \end{array}$$

$$1\times10^2 + 3\times10^1 + 7\times10^0$$

$$=\underline{137}.$$

✱ Decimal to Binary -

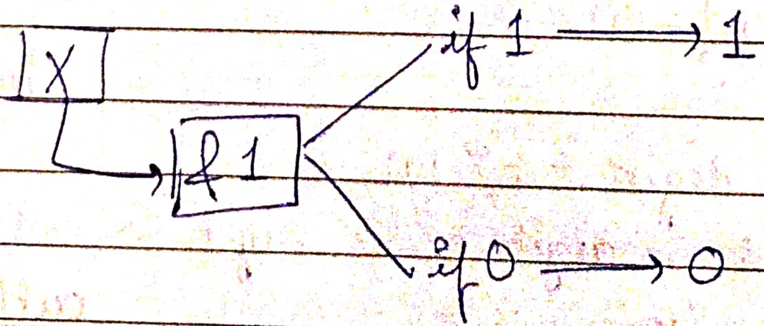
$$\left.\begin{array}{l} 5 \to 101 \\ 2 \to 10 \\ 6 \to 110 \end{array}\right\} \text{How?}$$

5 → X X X

5 ka system me koi binary ki form me store hua hoga.

As we know → 1 & 1 → 1
1 & 0 → 0

So bit ko AND karke pta chal jaega ki bit 1 hai ya 0 because if AND 1 karke 1 aya then it is 1 and if answer is 0 then bit is 0.



⇒ Binary Representation by logic -

```
while (n! = 0)
{
    int bit = n & 1;  //extract 1 bit.
    cout << "bit:" << bit << endl;
    ~~n=n>>1;~~ n=n>>1;  //checking next bit as
                         // right shift will destroy
                         // last bit.
}
```

4 → 1 0 0

→ 0   ans = (bit * 10^i) + ans.

→ 0   ans = 0

→ 1   i=0 → (0 × 10^0) + 0 = 0+0 = 0

⇒ Binary Rep. using formula:

i = 1 → (0 × 10^1) + 0 = 0 * 0 = 0

i = 2 → (1 × 10^2) + 0 = 100 + 0

= 100

```
#include <math.h>
int main()
{ int n=4;
  int i=0;
  int ans =0;
  while (n! =0)
   {
     int bit = n & 1;
     ans = (bit * pow (10,i)) + ans;
     n = n>>1;
     i++;
   }
  cout << "Binary representation:" << ans << endl;
}
```

output → Binary representation : 100

→ n%10 → last digit of no.

→ n&1 → last/rightmost bit

```
9 → binary
2 | 9
2 | 4 — 1  }
2 | 2 — 0  }  1001
2 | 1 — 0  }
  | 0 — 1  }
```

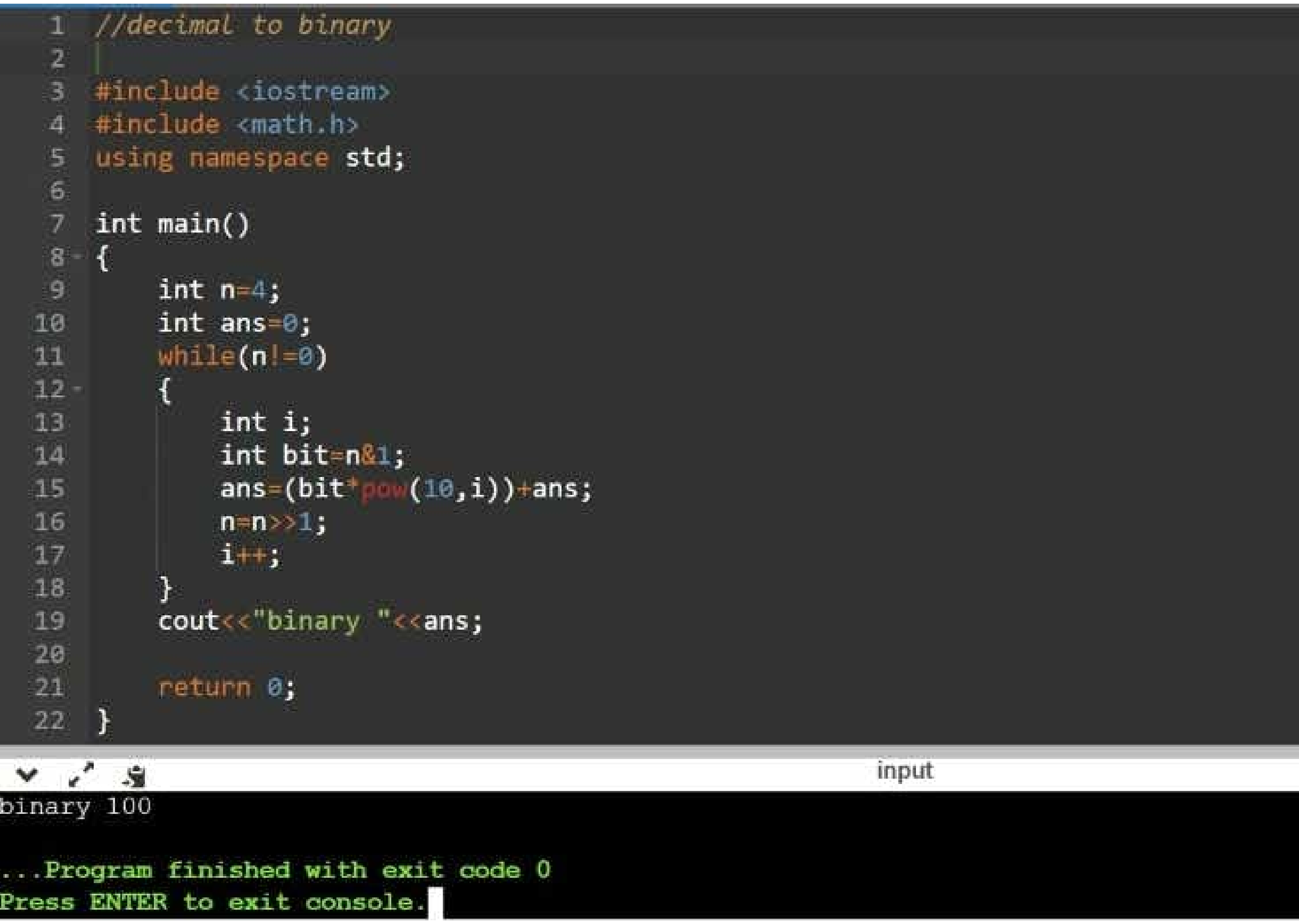
#H/W: Do prime Number Question by- Create an "i" outside for loop & remove bool.

even → &1 → 0
even → %2 → 0

odd → &1 → 1
odd → %2 → 1

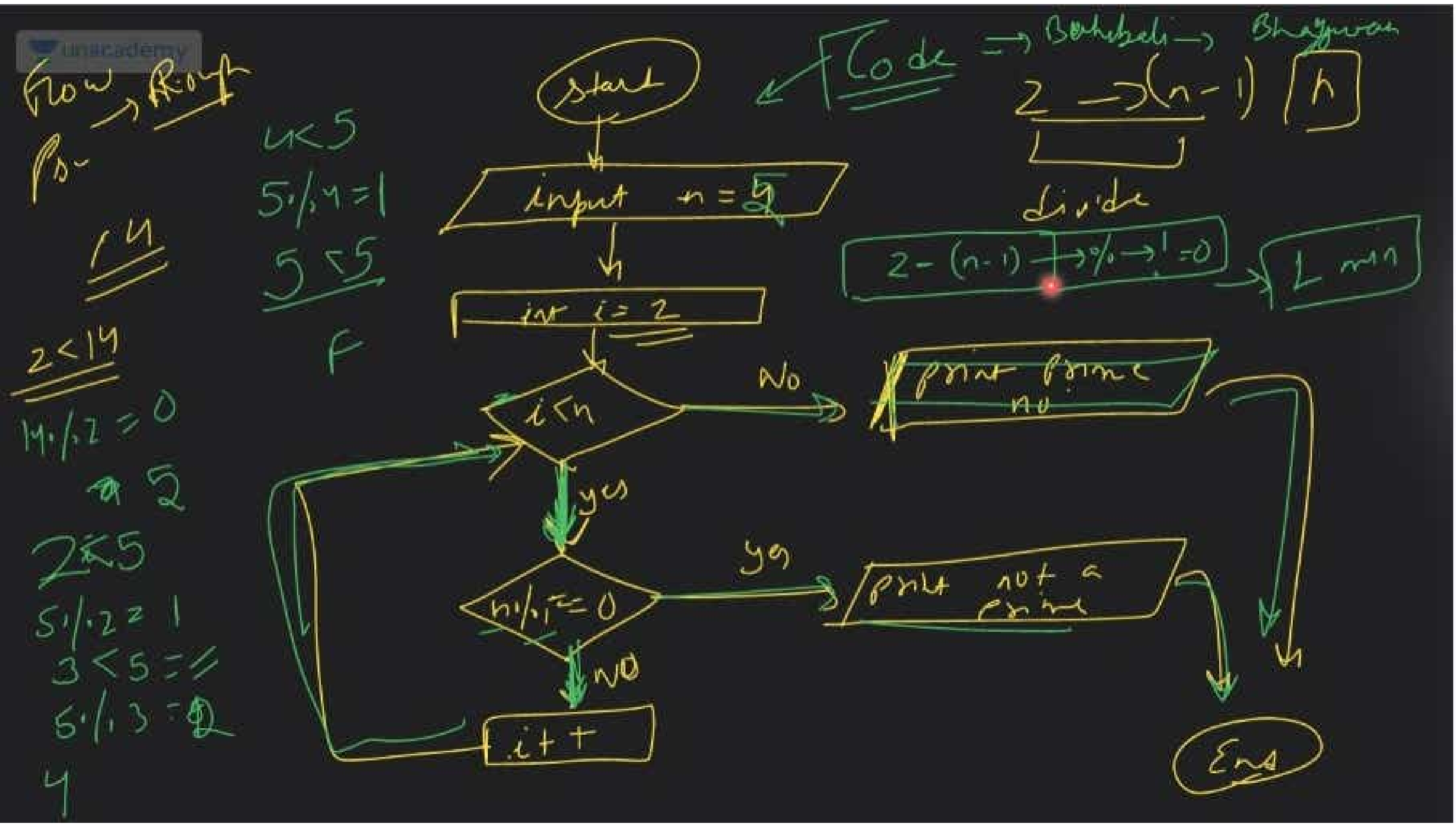
% is a heavy operation therefore its preferred to use & operator since it is not heavy. '&' is fast as it works on bit level.

H/W Binary to decimal } code → try for -ve no's.
Decimal to binary }



```cpp
//decimal to binary

#include <iostream>
#include <math.h>
using namespace std;

int main()
{
    int n=4;
    int ans=0;
    while(n!=0)
    {
        int i;
        int bit=n&1;
        ans=(bit*pow(10,i))+ans;
        n=n>>1;
        i++;
    }
    cout<<"binary "<<ans;

    return 0;
}
```

```
binary 100

...Program finished with exit code 0
Press ENTER to exit console.
```


Flow → Rough
Psu
4<5
5%4=1
5<5
F
14
2<14
14%2=0
2
2<5
5%2=1
3<5
5%3=2
4
Start
input n=5
int i=2
i<n
No
print prime no
yes
n%i==0
yes
print not a prime
NO
i++
End
Code → Bahubali → Bhagwan
2 → (n-1) n
divide
2-(n-1) → % → !=0 → 1 min