Date 03/10/2023
Page

while loop

while loop is similar to for loop. i.e while loop is alternative to for loop, People use as per their comfort.

e.g:-

1 to n

using for loop:

initialisation, condition to break loop, updation.

```
for (i=1; i<=10; i=i+1)
{
    cout<<i<<" ";
}
```

using while loop:

syntax:-

```
initilize
while(break)
{
  update
}
```

```
int i=1;
while(i<=10)
{ cout<<i<<" ";
  i=i+1;
}
```

i=1 2 3 4 ... 10

print 1 2 3 4 10

11 <= 10 → false → out of loop

★ Wap to print table using while loop.

n = 8 →

```
int i=1;
while(i<=10)
{
  cout<<n<<"x"<<i<<"="<<n*i<<endl;
  i=i+1;
}
```

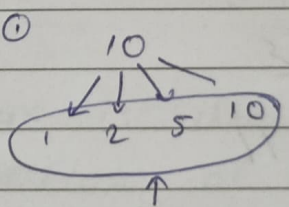
8 x 1 = 8
8 x 2 = 16
8 x 3 = 24
8 x 4 = 32
8 x 5 = 40
⋮
8 x 10 = 80

Refer code: 09-while and do-while loop in c++ / Class code / Table.cpp.

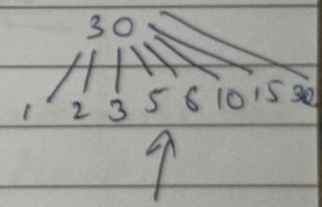
★ Wap to print factors of n.

Factors ← number that can divide n.

e.g



factors of 10.



factors of 30.

```
int i=1;
while(i<=n)
{
  if(n%i==0)
    cout<<i<<" ";
  i=i+1;
}
```

Refer code:.../factors.cpp

★ H/w odd / even → All 1 to n using while loop.

do-while loop

for / while loop	do-while loop
① initialize	① initialize
② update	② break condition
③ break condition	③ update
↑ in for/while loop	↑ in do-while loop.

In do-while loop first ~~print~~ execute the body then condition check, so even if condition is false initially then also loop will be executed once.

Syntax :-

```
do{
    //
    // update
}while (break);
```

e.g.:- print 1 to n

```
int i=1;
do{
    cout<<i<<" ";
    i++;
}while(i<=n);
```

n = 5

i=1 print 1 — 1<=5 true

i=2 print 2 — 2<=5 true

i=3

:

* WAP to print sum of n natural no.

n = 10

1 to ro

sum = 0

sum = sum+i;

i = 1

do Program :

Sum=0	
Sum=sum+i → 1	1
3	2
6	3
10	4
15	5
21	6
28	7
36	8
45	9
55	10

```
int i=1, sum=0;
do{
Sum=sum+i → Sum += i
    i++;
}while(i<=10)
cout<<sum;
```

Break, Continue

Break: Break is used to terminate the ~~loop~~ flow of program/loop.

Continue: Continue is used to skip the ~~current~~ current execution and continue with next itiration.

e.g.: Suppose we want a program that will print 1 to n, but when i==5, ~~loop~~ we don't want further to print anything.

```
i=1
while(i<=n)
{ if(i==5)
    break;
  cout<<i;
  i++;
}
```

op: 1 2 3 4

e.g Print 1 to n
if x no. then don't print further.

```
n=10
x=7
i=1
while(i<=n)
{
  if(i==x)
  {
    break;
  }
  cout<<i;
  i++;
}
```

continue:-

Skip current itiration.

e.g: Wap to print 1 to n, if n divide by 4 then skip the number.

```
~~i=1~~
~~for~~ for(i=1; i<=10; i++)
  { if(i%4==0)
      continue;
    cout<<i<<" ";
  }
```

op: 1 2 3 5 6 7 9 10

Switch

```
i = 1 -> Rohit
i = 2 -> Mohit
else  -> Sohit
```

->

```
if(i==1)
    cout<<"Rohit";
else if(i==2)
    cout<<"Mohit";
else
    cout<<"Sohit";
```

Above program can also be written using switch statement.

~~i==1~~

Syntax.

```
i=1
switch( i )        // i: variable
{
  case 1:
      // break;
  case 2;
      // break;
  :
  default:
      // break;
}
```

```
switch(i)
{
  case 1;
      cout<<"Rohit";
      break;
  case 2;
      cout<<"Mohit";
      break;
  default:
      cout<<"Sohit"
      break;
}
```

if we don't use break in case then "fall through" will happen. i.e all the ~~statement~~ below case will also be executed

Refer code : .../switch.cpp.

e.g:-

```
char name = ' ';
switch (name)
{
  case 'a':
        cout << "Hello";
        break;
  case 'b':
        cout << "Nope";
        break;
}
```

* default in switch is optional.
Refer code : ../switch.cpp

on case we can use int, char but can't use float or double type of value.

Scope of a variable

- Block start from '{' and end at '}'.
- scope of a variable is within block only (curly braces).

- within a same block duplicate variable name is not valid. but in case of child block duplicate variable name is valid.
child block ← block within block.

e.g (i)

```
int main()
{
  int num=10
  cout<<num; //10
  int num=20; X not valid.
}
```

(ii)

```
int main()
{
  int num=10
  cout<<num; //10
  if(n==10)
  {
    int num=20; ✓ valid.
    cout<<num; // 20
  }
  cout<<num; //10.
}
```

Here num's scope is within if block only ouside if block this num is not available.

O/P: 10 20 10.

Refer code: ../scope.cpp.

Date 03/10/2023
Page

Code For Decimal to Binary And Vice-Versa.

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
⋮	⋮

Base	Quotient	Rem
2	13	
2	6	1
2	3	0
2	1	1
	0	1 ↑

Decimal → Binary.
13 → 1101

Total no. of unique digit in a number system is known as base of that number system.

Binary
0, 1 → base 2

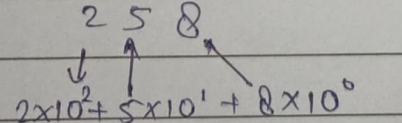
e.g.

a = 2, b = 5, c = 8

258

Ans = 2×10 + b
= 20 + 5 = 25
= 25×10 + 8
= 250 + 8 = 258

i.e



200 + 50 + 8 = 258.

e.g (11)

3 6 2 4 5 ← num

3

Ans → 3x10+6 = 36

Ans → 36x10+2 = 362

: 362x10+4 = 3624

3624x10+5 = 36245

```
int Ans=0;
Ans = Ans x10 + num
```

★ if a = 6, b = 9, make 96.

b x 10 + a

9 6

6492 → 2946

$2\times10^3 + 9\times10^2 + 4\times10^1 + 6\times10^0$

num

6 x 1+0 ⇒ 6 ← Ans

4x10+6 ⇒ 46 ← Ans

9x100+46 ⇒ 946 ← Ans

2x1000+946 ⇒ 2946 ← Ans

```
Ans=0
Ans = num x 10^i + ans
```

⇒ 2

num	Rem
13	
6	1
3	0
1	1
0	1

1101

Ans=0

Ans = rem x 10^i + ans

= 1 x 10^0 + 0 = 1

= 0 x 10^1 + 1 = 01

= 1 x 10^2 + 1 = 101

1 x 10^3 + 101 = 1101

```
int n =13
int rem, ans=0, mul=1;
while (n>0)
{
    rem = n%2;
    n = n/2;
    ans = rem x mul + ans
    mul = mul x 10;
}
cout << ans;
```

refer code: ../DecimalToBinary.cpp

Debug/Dry Run

n=13 13>0 True
Ans=0
mul=1

(i) rem = 13%2 = 1
ans = 1x1+0 = 1

(ii) n=6 6>0 true
ans=1
mul=10
rem = 6%2=0
ans = 0x10+1=1

(iii) n=3 3>0 True
ans=1
mul=100
rem = 3%2=1
ans = 1x100+1=101

(iv) n=1 1>0 True
ans=101
mul=1000
rem = 1%2=1
ans = 1x1000+101
= 1101

(v) n=0 0>0 false
n>0 false
↑
false
↑
loop break

Ans = 1101

Binary To Decimal

101

↳ $1\times2^2 + 0\times2^1 + 1\times2^0$

$4 + 0 + 1 = 5$

1101

↳ $1\times2^3 + 1\times2^2 + 0 + 2^1 + 1\times2^0$

$8 + 4 + 0 + 1 = 13$

ans = rem x 2^i + ans

rem = num%10;
num = num/10;

$1\times2^0 + 0 = 1$

$0\times2^1 + 1 = 1$

$1\times2^2 + 1 = 5$

$1\times2^3 + 5 = 13$

10	1101	rem
10	110	1
10	11	0
10	1	1
10	0	1

```
int num = 1101 ;
int ans = 0, rem, mul = 1;
while (num > 0)
{
    rem = num%10;
    num = num/10;
    ans = rem x mul * ans
    mum *= 2 ;
}
cout << ans;
```

Refer code: ../BinaryToDecimal.cpp

Dry Run

(i) n = 1101
1101 > 0
true
rem = 1101%10
= 1
num = 110
ans = 1x1 + 0
= 1
mul = 2

(ii) n = 110
110 > 0
true
rem = 110%10
= 0
num = 11
ans = 0x2 + 1
= 1
mul = 4

(iii) n = 11
11 > 0
true
rem = 11%10 = 1
num = 1
ans = 1x4 + 1 = 5
mul = 8

(iv) n = 1
1 > 0
true
rem = 1%10 = 1
num = 0
ans = 1x8 + 5
= (13)
mul = 16

(v) n = 0
0 > 0
false.

aditya
Date
Page

Decimal to octa

8	13	
8	1	5
	0	1 ↑

$(13)_{10} \Rightarrow (15)_8$

octa to Decimal

$(15)_8$

$1 \times 8^1 + 5 \times 8^0$

$8 + 5 = (13)_{10}$

Binary To octa

$(1101)_2 \longrightarrow (\quad)_8$

$(1101)_2 \longrightarrow (\quad)_{10}$

$(\quad)_{10} \longrightarrow (\quad)_8$

Octa to Binary

$(\quad)_8 \longrightarrow (\quad)_2$

$(\quad)_8 \longrightarrow (\quad)_{10}$

$(\quad)_{10} \longrightarrow (\quad)_2$

Leetcode Problem

(1) Add digit (#258)

e.g. (i) 27

$2+7 = 9$

(ii)

10	62	
10	6	2
	0	6

Ans = 0 $2+6 = 8$

Ans = rem + ans

$= 2+0 = 2$

$8+2 = (8)$.

(iii) 249

$2+4+9 = 17$

$= 8$

we need single digit ans only.

```
int num;
while(num > 9)
{ int num, ans=0, rem;
  while(num != 0)
  {
     rem = num % 10;
     num = num / 10;
     ans = ans + rem;
  }
     num = ans;
}
cout << ans;
```

(2) Leap year (4F4)

1600 ✓
1700 X
1800 X
1900 X
2000 ✓
2004 ✓

(i) year which divides by 400 then that year is leap year.

(ii) year which divides by 4 but not with 100. is also leap year.

```
int year
if( year % 400 == 0)
        cout << "Leap year";
else if ( year % 4 == 0 && year % 100 != 0)
        cout << "Leap year";
else
        cout << "Not leap year";
```

(3) Reverse Integer

(i) i/p = 234
o/p = 432

(ii) i/p 476
o/p 674

rem = 234 % 10 = 4

ans = 0 ~~rem + ans~~
ans = ans x 10 + rem
= 0 x 10 + 4 = 4
= 4 x 10 + 3 = 43
= 43 x 10 + 2 = 432.

```
int ans=0, rem;
while(num>0)
{
int ans=0, rem;
  rem = num%10;   ← last digit.
  num = num/10;
  ans = ans*10+rem;
}
cout<<ans;
```

(i) integer overflow

INT_MAX

INT_MIN

$ans*10 + rem > INT\_MAX$

$ans > \frac{INT\_MAX - rem}{10}$

$ans > \frac{INT\_MAX}{10}$

if $(ans > \frac{INT\_MAX}{10})$

return 0;

(ii)

$ans*10 + rem < INT\_MIN$

if $(ans < \frac{INT\_MIN}{10})$

return 0;

(4) Power of 2

$2^1 \rightarrow 2$ yes
$2^2 \rightarrow 4$ yes
8 yes

15 → NO

Num<1 ← NO ← Negative number.

(i)

1 — 1 ← 2^0
2 — 10 ← 2^1
4 — 100 ← 2^2
8 — 1000 ← 2^3
16 — 10000 ← 2^4
32 — 100000 ← 2^5

(ii)

2 31
4
8
16 64
32

divide num by Power of 2 till num exided or reached that num.

```
while (x != 1)
{
    if(x % 2 == 1)
        return 0;
    x /= 2;
}
return 1;
```

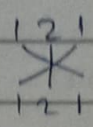
(5) sqrt x

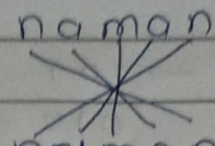
H/W

4 → √16

√25 → 5

(6) Pallindrome





same.

63 → ← 36 ✗

```
int x, rem.
ans = 0
num = x;
while (num != 0)
{
    rem = num % 10;
    num = num / 10;
    ans = ans * 10 + rem;
}
if ( x == ans)
    return 1;

if( ans > INT_MAX / 10 || ans < INT_MIN / 10 )
    return 0;
```

① Complement of a no:

5 → 101

010 = (2) ✗.

13 → 1101

0010 = (2) ✗

27 → 11011

00100 = (4) ✗.

2	27	rem		
2	13	1	2^0	0
2	6	1	2^1	0
2	3	0	2^2	4
2	1	1	2^3	0
	0	1	2^4	0
				(4)

2	13	rem		
2	6	1	2^0	0
2	3	0	2^1	2
2	1	1	2^2	0
	0	1	2^3	0
				(2)

```
ans = 0, rem, mul = 1
while (n)
{
    rem = n % 2 ;
    n /= 2;
    rem = rem ^ 1;
    ans = ans + rem x mul;
    mul *= 2 ;
}
return ans ;
```

0 ^ 1 = 1
1 ^ 1 = 0

edge cases
n = 0 , return 1

```
if( n == 0)
    return 1;
```

Date 05/10/2023
Page

Function in C++

Problem statement

let
a = 6
b = 8

suppose we have to calculate
(i) prime or not for a
(ii) factorial of a
(iii) prime or not for b
(iv) factorial of b
(v) prime no b-a
(vi) factorial b-a

```
int main()
{
  int a, b;
  cin>>a>>b;
  [prime] - a
  [factorial] - a
  [prime] - b
  [factorial] b
  :
}
```

In above problem, for, prime no. we have to write 3 times with different number, in the same way we have to write factorial program three times. for the same work we have to write same code for multiple times. so here function came into picture.

Function :

function says don't write some piece of code multiple times, write once and reuse it when ever required.

- Reusibility.
- Code Readability.

Syntax :

```
return_type function_name (parameters)
{
    // code

    return value;
}
```

Return Type

- int
- float
- char
- double
- void

eg

```
bool prime(int n)
{ if(n<2)
    return 0;
  for(i=2; i<n; i++)
  {
    if(n%i==0)
      return 0;
  }
  return 1;
}
```

Argument ↓ function

```
int factorial (int n) // declaration
{
    int ans = 1;
    for(int i = 1; i <= n; i++)
        ans = ans * i;
    return ans;
}
```

function defination

```
int main()
{
    int a, b;
    cin >> a >> b;
    cout << prime(a);
    cout << factorial(a);
    cout << Prime(b);
    cout << factorial(b);
    cout << prime(b-a);
    cout << factorial(b-a);
}
```

function call

e.g (11)

```
int Sum(int a, int b)
{
    int ans = a+b;
    return ans;
}
int main()
{   int m=4, n=5;
    cout << sum(m, n);
}
```

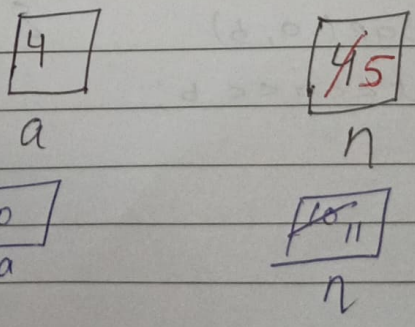
```
int fact(int n=3) // Default parameter.
{
    // code
}
```

in case of function call of default parameterized function. if you we pass some parameter during function call then that function will work on passed parameter, but if we do not pass any parameter that that function will do operation on default parameter which was given during function declaration/definition.

pass by value

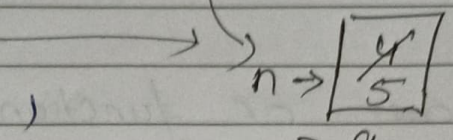
```
int main()
{ int a = 10;
  incr(a);
  cout<< a; // a=10
}
```

```
void incr(int n)
{
  n++);
}
```

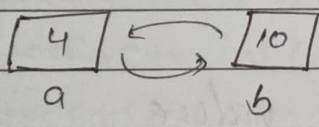


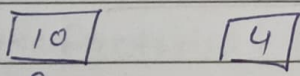
pass by reference

```
int incr(int &n)
{
    n++;
}
int main()
{
  int a = 4;
  incr(a);
  cout<<a; //5
}
```



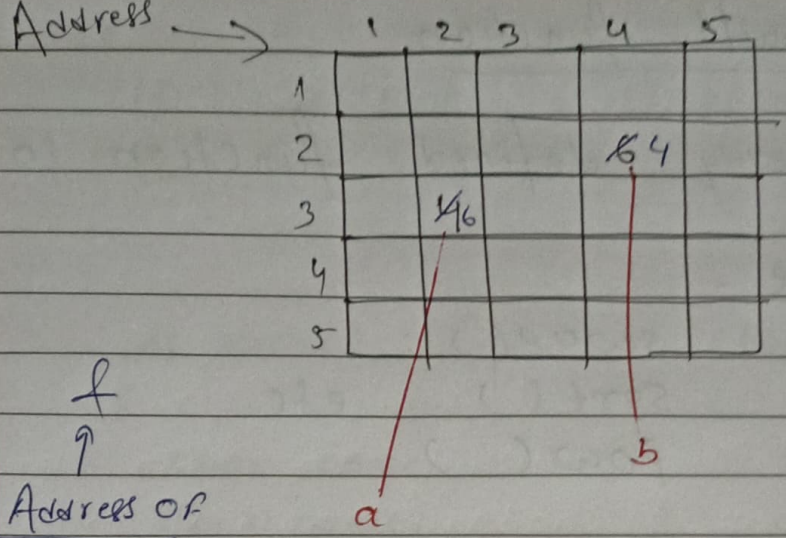
swap






```
int main()
{
  int a, b;
  cin>>a>>b;
  swap(a, b)
  cout<<a<<b;
}
```

```
// call by value
swap(int a, int b)
{
    int c;
    c = a;
    a = b;
    b = c;
} // will not swap.
// pass by reference.
swap(int &a, int &b)
{ int c;
  c = a;
  a = b
```



& → Address of.

function overloading

function with same name but with different parameter.

```
void swap (int &a, int &b)
{


}
void swap(float &x, float &y)
{


}.
```

Inbuilt. function.

Already defined function in c++.

Like -

swap( );
sqrt( ) .... etc.
pow( ) ..

WAP to convert 'a'-'z' to 'A'-'z'.

I/P : {a, b, c, ... }

O/P : { A, B, C, D .... }

```
int main()
{
    char name;
    cin >> name;
    cout << convert(name);
}
```

```
char convert(char ch)
{
    char ans;
    ans = ch - 'a' + 'A';
    return ans;
}
```

(1)

```
ans = name - 'a' + 'A';
```

A → 65

a → 97

let

name = c

'c' - 'a' + 'A';

2 + 'A';

ans = C;

Armstrong no.

eg (1)

23

2 digit

no. of digit

$2^2 + 3^2 = 4 + 9 = 13$

23 == 13

23 is not Armstrong no.

if yes → armstrong no.

No - Not Armstrong no.

(11) 153

3 digit

$1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$

153 == 153

↓

yes ∴ 153 is Armstrong No.

```
int countDigit(int num)
{
  int count = 0;
  while(num)
  {
    count++;
    n /= 10;
  }
  return count;
}

int main()
{
  int n;
  cin>>n;
  int digit=countDigit(n);
  cout<<Armstrong(n,digit);
}

bool Armstrong(int num, int digit)
{
    int n = num, ans = 0, rem;
  while(n)
  { rem = n%10;
    n /= 10;
  ans+=pow(rem, digit) // (rem^digit).
  }
  if(ans == num)
      return 1;
  else
      return 0;
}
```

Find Trailing zero in a fact. (Ufq)

O/P

e.g $6! = 720$ → 1

$8! = 40320$ → 1

(I) $6! = 1 \times 2 \times 3 \times 4 \times 5 \times 6$

$2^2 \quad 2 \times 3$

$2 \times 3 \times 2^2 \times 5 \times 2 \times 3$

$2^4 \times 3^2 \times 5$

$5 \times 2 = 10$

(II) $2^4 \times 3^2 \times 5^2$

$2^2 \times 5^2 = 100$ → 10

5×2

No. of 2's and 5's

(III) $2^4 \times 5^3 = 10^3$

↳ 000

No. of 5 if n!.

$2^n \times 3^4 \times 5^2$

$n > 4 > 2$ →

$1 \times 2 \times 3 \times 4 \times 5 \times \ldots \times 10 \times \ldots \times 15 \times \ldots \times 20$

! $5/5 = 1$! ! $15/5 = 3$!

$\times \ldots 25 \ldots$ 50

5×5 $5 \times 5 \times 2$

(2)

$\frac{25}{5} = \frac{5}{5} = 1$ $\frac{50}{5} = \frac{10}{5} = 2$

$5 + 1 = 6$. $10 + 2 = 12$.

$100! \rightarrow \frac{100}{5} = \frac{20}{5} : 4$

$20 + 4 = (24)$

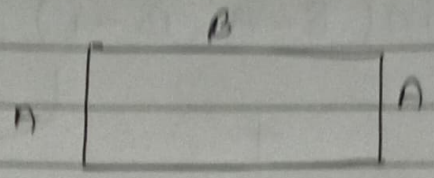
$\underline{148!} \Rightarrow \frac{148}{5} = \frac{29}{5} + \frac{5}{5} = 1$

$29 + 5 + 1 = (35)$.

```
int trailingzero(int n)
{
    int count=0;
    while(n>=5)
    {
        count += n/5;
        n /=5;
    }
    return count;
}
```

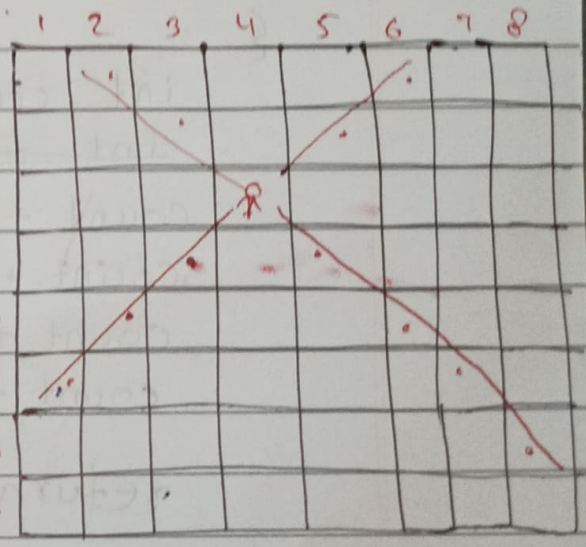
Rectangle (InterviewBit)

a, b, c, d




```
bool Reactangle (int a, int b, int c, int d)
{
    if ((a == b) && (c == d) ||
        (a == c) && (b == d) ||
        (a == d) && (b == c) )

        return 1;

    else

        return 0;

}
```

Bishop (InterviewBit)



A B

3 4 (11)

Total : 8 4

(R, D) 8 8

Right, Down 3 4

(5, 4)

min (4)

8, 1

3 4

(5, 3) → (3) (Left, Down)

min(8-A, 8-B)
↳ Right, Down.
+
min(8-A, B-1)
↳ (Left, Down)
+
min(A-1, B-1)
↳ (left, UP)
+
min(A-1, 8-B)
↳ (Right, UP)

Total = min(8-A, 8-B) + min(8-A, B-1)
+ min(A-1, B-1) +
min(A-1, 8-B).

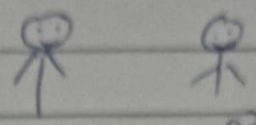
```
int Bishop( int A, int B)
{
    int count = 0
    int ans = max(2,3
    count += min(8-A, 8-B);
    count += min(8-A, B-1);
    count += min(A-1, B-1);
    count += min(A-1, 8-B);

    return count;
}
```

Nim Game (leetcode)

We can take 1 to 3 step.

1 2 3

n



1st 2nd

1 2 3

4 5 6 3, 4

1st make always factor of 4, I will always win, but if no. is already in 4 factor then I will loose.

n = 4, 8, 12, 16, 20

↳ loose.

```
if (n % 4 != 0)
    return 1; // win
else
    return 0; // loose.
```