CODE FOUNDATION

LESSON 02

Operators in C++:

Operator is basically a symbol that is given to an operation to operate on some value. It tells the computer to perform some mathematical or logical calculations. For example, **+** is an operator that is used to add two numbers. Operand is the numerical value on which we have to perform operation.
For e.g In 3+2, + is an operator and it performs addition operation on 3 and 2, so 3,2 are operands.

Classification of operators based on their usage:

1. Assignment operators
2. Arithmetic operators
3. Relational operators
4. Logical operators
5. Bitwise operators
6. Increment/Decrement operators
7. Ternary operators
8. Other operators

Let's discuss the operators one-by-one:

1.Assignment operator(=): Assignment operator is used to assign a value to a variable(or identifier).Equal sign(=) is the assignment operator.
For e.g.:

```
int x, y;  // declaring 2 variables of type int
    x = 10;    // assigning the value 10 to x
    y = 20;    // assigning the value 20 to y
```

Here, we have assigned the values 10 and 20 to the variables x and y respectively. It is important to note here that the value that we want to assign to a variable, is always given on the right-hand side of "=" and the variable is always written on the left-hand side of "=".

Let's take one more example to illustrate it:

```
int x, y, z;
    x = y = z = 30;
```

Here we have declared three variables, namely *x*, *y* and *z*, in the first statement. In the next statement, the value *30* is first assigned to *z*, then the value in *z* is assigned to *y* and finally the value of *y* is assigned to *x*.

Thus, the ***assignment operator(=)*** has ***right to left associativity***, i.e., it assigns the value given on its right-side to the variable written on its left-hand side.

2.Arithmetic operators(+, -, *, /, %): Arithmetic operators are used to perform simple mathematical operations such as addition, subtraction, multiplication, division and modular division.
The following operators are supported by C++:

OPERATOR	DESCRIPTION
+	Adds two operands
-	Subtracts second operand from the first one
*	Multiplies the two operands
/	Divides first operand from second one
%	Modulo(Evaluates remainder when first operand is divided by the second one)

Let's take a code snippet to understand the arithmetic operators:

```
int x, y;
    x = 10;      // x = 10
    y = 20;      // y = 20
    x = x + y;   // x = 30 (10 + 20)
    y = x - y;   // y = 10 (30 - 20)
    x = x - y;   // x = 20 (30 - 10)

    x = x * y;   // x = 200 (20*10)
    y = x / y;   // y = 20  (200 / 10)
    x = x / y;   // x = 10  (200 / 20)

    int z;
    z = x % y;   // z = 10 (Remainder when 10 is divided by 20 is 10)
```

Note: The modulo operator(%) cannot be applied on a float and the sign of the remainder is the same as that of the numerator.

These were all the examples of binary operators: Binary operators are those operators which work on two operands.
There are unary arithmetic operators also, like increment(++) and decrement(--). We will study about them later.

In the above examples we saw the statements having only one operator. Now the question arises how to calculate the values when there are more than one operators?

Operator Precedence:

Operator precedence defines the order in which we operations are evaluated when there are more than one operand.
Let's have a look at an example
Here is an expression: **4+3*9-2**
How will you evaluate this expression? There can be many ways to do this.

Suppose you first evaluate 4+3=7, then multiply 7 and 9 which evaluates to 63 and then subtract 2 from 63 which will give the final result as 61.
But if you multiply 3 and 9 first and then perform the addition of 27 and 4 which evaluates to 31 and finally subtract 2 from 31 which gives the result as 29.
Did you notice we got different results for the same expression?

Now you must be clear that we need a rule which will define the order in which the operations are evaluated.

Precedence of arithmetic operators:

Operator	Relative precedence
++, --	1 Highest precedence
*, /, %	2
+, -	3 Lowest precedence

Note: If the precedence level of operators is the same in an expression, then we operators are performed in left-to-right manner and if there are

any parenthesis in an expression, ten we first need to evaluate the parentheses.

If we now look at the above example expression **4+3*9-2**
We now know that multiplication operation will be performed first. Since + and - have the same precedence we will perform the operations in left to right manner. The correct value of this expression is 29.
Note: In case of many operators in a single expression it can sometimes become difficult to find the precedence order. In these cases you should use parentheses to make the order of evaluation clear.

3. Relational Operators(==,>=,<=):

To compare the values of operands, we use relational operators. If we have to check if two operands are equal or not? or Is operand1 greater than the second operand? we use relational operators.

Operator name	Symbol	Example	Questions
Less than	<	2<8	Is 2 less than 8? Here the result is false.
Greater than	>	6>3	Is 6 greater than 3? Here the result is true.
Less than or equal to	<=	3<=4	Is 3 less than or equal to 4? Here the result is true. Also notice 3<=3 will also be true.
Greater than or equal to	>=	8>=10	Is 8 greater than or equal to 10? Here the result is true.
Equal to	==	5==7	Is 5 equal to 7? Here the result is false. 6==6 will give a true result.
Not equal to	!=	5!=9	Is 5 not equal to 9? Here the result is true.

Note: '=' is an assignment operator, it assigns the value on its right side to a variable on its left side. Don't confuse it with '=='. '==' is a relational operator. It checks whether the two operands are true or not.

4. Logical operators:

Logical operators are used to determine the logic between variables or values:

Operator	Name	Description	Example
&&	Logical and	Returns true if both expressions are true	x < 5 && x < 10 Here first expression is x<5 and second expression is x<10
||	Logical or	Returns true if one of the expression is true	x < 5 || x < 4
!	Logical not	Reverses the result, returns false if the result is true	!(x < 5 && x < 10) 5!=4

Example :-

```
#include <iostream>
using namespace std;
int main(){
        bool b1 = true;
        bool b2 = false;
        cout<< "b1 && b2: " << (b1 && b2) << endl;
        cout<<"b1 || b2: " << (b1 || b2) << endl;
        cout<<"!(b1 && b2): " << !(b1 && b2);
        return 0;
}
```

Program Link(Sapphire Engine): https://sapphireengine.com/@/8bm8yr

Output:

```
b1 && b2: 0
b1 || b2: 1
!(b1 && b2): 1
```

Important note:

1. **In case of logical and(&&) second expression is evaluated only if the first expression is true. If the first expression is false then the second expression is not evaluated.**
2. **In case of logical or(||) the second expression is not evaluated if the first expression is true.**

For eg int a=4,b=2;

```
bool value=((a<b) && cout<<"Second expression");
```

Since a<b evaluates to false so the second expression is not even evaluated thus the above code snippet gives no output.

But, int a=4,b=2;

```
bool value=((a>b) && cout<<"Second expression");
```

This code snippet gives output as: Second expression

It means when the first expression is true then only the second expression is evaluated in case of logical and(&&).

5. Bitwise operators:

There are six bitwise Operators: & , | , ^ , ~ , << , >>

```
num1 = 11;          /* equal to 00001011*/
num2 = 22;          /* equal to 00010110 */
```

Bitwise operator performs bit by bit processing.

num1 & num2:

compares corresponding bits of num1 and num2 and generates 1 if both bits are equal, else it returns 0. In our case it would return: 2 which is 00000010 because in the binary form of num1 and num2 only second last bits are matching.

num1 | num2:

compares corresponding bits of num1 and num2 and generates 1 if either bit is 1, else it returns 0. In our case it would return 31 which is 00011111

num1 ^ num2:

compares corresponding bits of num1 and num2 and generates 1 if they are not equal, else it returns 0. In our example it would return 29 which is equivalent to 00011101

~num1:

is a complement operator that just changes the bit from 0 to 1 and 1 to 0. In our example it would return -12 which is signed 8 bit equivalent to 11110100

num1 << 2:

is a left shift operator that moves the bits to the left, discards the far left bit, and assigns the rightmost bit a value of 0. In our case output is 44 which is equivalent to 00101100

Note: In the example below we are providing 2 at the right side of this shift operator that is the reason bits are moving two places to the left side. We can change this number and bits would be moved by the number of bits specified on the right side of the operator. Same applies to the right side operator.

num1 >> 2:

is a right shift operator that moves the bits to the right, discards the far right bit, and assigns the leftmost bit a value of 0. In our case output is 2 which is equivalent to 00000010

Program Link(Sapphire Engine): https://sapphireengine.com/@/xsw6z5

```
1. #include <iostream>
2. using namespace std;
3. int main(){
4.     int num1 = 11;  /* 11 = 00001011 */
5.     int num2 = 22;  /* 22 = 00010110 */
6.     int result = 0;
7.     result = num1 & num2;
8.     cout<< "num1 & num2: " << result << endl;
```

```
9.     result = num1 | num2;
10.       cout<< "num1 | num2: " << result << endl;
11.       result = num1 ^ num2;
12.       cout<< "num1 ^ num2: " << result << endl;
13.       result = ~num1;
14.       cout<< "~num1: " << result << endl;
15.       result = num1 << 2;
16.       cout<< "num1 << 2: " << result << endl;
17.       result = num1 >> 2;
18.       cout<< "num1 >> 2: "<<result;
19.       return 0;
20.   }
```

Output:

num1 & num2: 2
num1 | num2: 31
num1 ^ num2: 29
~num1: -12
num1 << 2: 44
num1 >> 2: 2

6. Increment/Decrement operators:

These are unary operators used to add 1 or subtract 1 from the operand.
++ is the increment operator. It adds 1 to the operand.
-- is the decrement operator. It subtracts 1 from the operator.
E.g **c++ & ++c has the same meaning as c=c+1**
and c-- & --c has the same meaning as c=c-1

There are two types of increment/decrement operators: Prefix and postfix. In prefix, operators are prefixed or placed before the variable. ++a is called pre-increment and --a is called pre-decrement. Similarly a++ is called post-increment and a-- is called post-decrement.

Operator	Called as	Example	Description
++	Pre-increment	++a	First increment the value of a, then use the new value of a in the expression in which a resides.
++	Post-increment	a++	Use current value of a in the expression in which a resides and then increment a.
--	Pre-decrement	--a	First decrement a by one and then use the new value of a in the expression in which a lies.
--	Post-decrement	a--	Use the current value of a in the expression in which a resides and then decrement a by one.

Example 1

Program Link(Sapphire Engine): https://sapphireengine.com/@/vqp675

Program Link(Ideone): https://ideone.com/JScUKJ

```
1. #include <iostream>
2. using namespace std;
3.
4. int main() {
5.      int a=3;
6.      //Pre-increment
7.      //This will first increment the value of a by 1 and
   then display the incremented value.
8.      cout<<++a<<endl;
9.      cout<<"Let's check the value of a: "<<a<<endl;
10.     return 0;
11.  }
```

Output

4

Let's check the value of a: 4

Note that here the value of a is incremented first. When we display the value of variable a, it shows that it's value is incremented.

Example 2

Program Link(Sapphire Engine):https://sapphireengine.com/@/gck4xd

Program Link(Ideone): https://ideone.com/zG7xyf

```
1. #include <iostream>
2. using namespace std;
3.
4. int main() {
5.      int a=3;
6.      //Post increment
7. //Here the initial value of a will be displayed first and
   then there will be increment.
8.      cout<<a++<<endl;
9.      cout<<"Let's check the value of a: "<<a<<endl;
10.     return 0;
11.  }
```

OUTPUT

3

Let's check the value of a: 4

In this example, notice that when we display the value of a++ the output is 3, which means that it displays the initial value of and then increments it's value.

7. Ternary operator:

?: is called a ternary operator. It uses three operands. The syntax is

variable= Expression1 ? Expression2 : Expression3

If Expression1 evaluates to true then Expression2 is executed and if Expression1 evaluates to false then Expression3 is executed.

Example

Program Link(Sapphire Engine):https://sapphireengine.com/@/cthhi5

Program Link(Ideone): https://ideone.com/VsbF6m

```
#include <iostream>
using namespace std;

int main() {
      // your code goes here
      int a=10,b=20;
      (a>b)?cout<<"a is greater than b"<<endl:cout<<"a is less than b"<<endl;
      return 0;
}
```

OUTPUT

a is less than b

In the above example, expression1 is (a>b), expression2 is cout<<"a is greater than b"<<endl and expression3 is cout<<"a is less than b"<<endl;
We check for expression1, if it is true then expression2 is executed and if expression1 evaluates to false then expression3 is executed.

8. Other operators:

There are other operators as well which can not be clearly classified in the above categories. Some of these operators are sizeof, . etc.

There is one more type of operator called the **compound assignment operators**.

Compound assignment operators:

In C++ we have shortcut method of combining binary arithmetic operations with the assignment operation.

For E.g a=a+6;

This statement can also be written as a+=6;

This means increment the value of a by 6 and then assign the incremented value of a to a;

General way of writing compound assignment operators:

Expression1 operator= Expression2;

This is equivalent to:
Expression1=Expression1 operand Expression2;

Simple assignment	Equivalent compound assignment	Example
a=a+b	a+=b	i+=3
a=a-b	a-=b	i-=6
a=a/b	a/=b	i/=7
a=a*b	a*=b	i*=4
a=a%b	a%=b	i%=3

Operator Precedence

Precedence	Operator	Description	Associativity
1	::	Scope Resolution	Left to Right
2	a++	Suffix/postfix increment	Left to Right
	a--	Suffix/postfix decrement	
	type( )	Function cast	
	type{ }	Function cast	
	a( )	Function call	
	a[ ]	Subscript	

	.	Member access from an object	
	->	Member access from object ptr	
3	++a	Prefix increment	Right to Left
	--a	Prefix decrement	
	+a	Unary plus	
	-a	Unary minus	
	!	Logical NOT	
	~	Bitwise NOT	
	(type)	C style cast	
	*a	Indirection (dereference)	
	&a	Address-of	
	sizeof	Size-of	
	co_await	await-expression	
	new new[ ]	Dynamic memory allocation	
	delete delete[]	Dynamic memory deallocation	
4	.*	Member object selector	Left to Right
	->*	Member pointer selector	

5	a * b a / b a % b	Multiplication Division Modulus [gives remainder when a is divided by b]	Left to Right
6	a + b a - b	Addition Subtraction	Left to Right
7	<< >>	Bitwise left shift Bitwise right shift	Left to Right
8	<=<	Three-way comparison operator	Left to Right
9	< <= > >=	Less than Less than or equal to Greater than Greater than or equal to	Left to Right
10	== !=	Equal to Not equal to	Left to Right
11	&	Bitwise AND	Left to Right
12	^	Bitwise XOR	Left to Right

13	|	Bitwise OR	Left to Right
14	&&	Logical AND	Left to Right
15	||	Logical OR	Left to Right
16	a ? b : c	Ternary Conditional	Right to Left
	throw	throw operator	
	co_yield	yield expression (C++ 20)	
	=	Assignment	
	+=	Addition Assignment	
	-=	Subtraction Assignment	
	*=	Multiplication Assignment	
	/=	Division Assignment	
	%=	Modulus Assignment	
	<<=	Bitwise Shift Left Assignment	
	>>=	Bitwise Shift Right Assignment	
	&=	Bitwise AND Assignment	
	^=	Bitwise XOR Assignment	
	|=	Bitwise OR Assignment	
17	,	Comma operator	Left to Right

CONDITIONAL STATEMENTS:

C++ has the following conditional statements:

- Use if to specify a block of code to be executed, if a specified condition is true
- Use else to specify a block of code to be executed, if the same condition is false
- Use else if to specify a new condition to test, if the first condition is false
- Use switch to specify many alternative blocks of code to be executed

Use the if statement to specify a block of C++ code to be executed if a condition is true.

Syntax

```
if (condition) {
  // block of code to be executed if the condition is true
}
```

Note that if is in lowercase letters. Uppercase letters (If or IF) will generate an error.

In the example below, we test two values to find out if 20 is greater than 18. If the condition is true, print some text:

Example

```
if (20 > 18) {
  cout << "20 is greater than 18";
}
```

We can also test variables:

Example

```
int x = 20;

int y = 18;

if (x > y) {

  cout << "x is greater than y";

}
```

Output :

Case1 - 20 is greater than 18

Case2 - x is greater than y

The else Statement

Use the else statement to specify a block of code to be executed if the condition is false.

Syntax

```
if (condition) {

  // block of code to be executed if the condition is true

} else {

  // block of code to be executed if the condition is false

}
```

Example

```
int time = 20;

if (time < 18) {

  cout << "Good day.";

} else {
```

```
  cout << "Good evening.";
}
// Outputs "Good evening."
```

Example explained

In the example above, time (20) is greater than 18, so the condition is false. Because of this, we move on to the else condition and print to the screen "Good evening". If the time was less than 18, the program would print "Good day" but in our case time is greater than 18 so the boolean statement (time < 18) becomes false and control shifts to else condition.

The else if Statement

Use the else if statement to specify a new condition if the first condition is false.

Syntax

```
if (condition1) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
  // block of code to be executed if the condition1 is false and condition2 is true
} else {
  // block of code to be executed if the condition1 is false and condition2 is false
}
```

Example

```
int time = 22;
if (time < 10) {
  cout << "Good morning.";
} else if (time < 20) {
```

```
  cout << "Good day.";
} else {
  cout << "Good evening.";
}
// Outputs "Good evening."
```

Example explained

In the example above, time (22) is greater than 10, so the **first condition** is false. The next condition, in the else if statement, is also false, so we move on to the else condition since **condition1** and **condition2** is both false - and print to the screen "Good evening".

However, if the time was 14, our program would print "Good day."

MORE EXAMPLES:

Example 1:

Program link(IDEONE): https://ideone.com/ckaus/ifelseexample

Program link(Sapphire Engine): https://sapphireengine.com/@/22ij5s

```
1. // Demonstration of logical operators
2. #include <iostream>
3. using namespace std;
4.
5. int main() {
6.
7.    double weight, height;
8.    // Enter your weight and height(in metres)
9.    cout << "Enter your weight and height(in metres): ";
10.      cin >> weight >> height;
11.
12.      // computing BMI(Body mass index)
13.      double bmi = weight / (height * height);
14.
15.      cout << "Your BMI is: " << bmi << endl;
```

```
16.
17.     // reporting your BMI category
18.     cout << "Your BMI category is: ";
19.     if (bmi < 15)
20.         cout << "Starvation" << endl;
21.     else if (bmi >= 15 && bmi < 17.5)
22.         cout << "Anorexic" << endl;
23.     else if (bmi >= 17.5 && bmi < 18.5)
24.         cout << "Underweight" << endl;
25.     else if (bmi >= 18.5 && bmi < 24.9)
26.         cout << "Ideal" << endl;
27.     else if (bmi >= 24.9 && bmi < 25.9)
28.         cout << "Overweight" << endl;
29.     else if (bmi >= 25.9 && bmi < 30.9)
30.         cout << "Obese" << endl;
31.     else
32.         cout << "Morbidly Obese" << endl;
33.     return 0;
34. }
```

Standard Output

```
Enter your weight and height(in metres):
62
1.72
Your BMI is: 20.9573
Your BMI category is: Ideal
```

Example 2:

Program link(IDEONE): https://ideone.com/ckaus/operatorsbasicprg

Program link(Sapphire Engine): https://sapphireengine.com/@/6zgm5c

```
1. #include <iostream>
2. using namespace std;
```

```
3.
4.  int main() {
5.
6.      int a = 1, b = 3, c;
7.              c = b <<  a;          // c = 3 << 1 (0011 << 1 = 0110) = 6
8.      cout << "Initial values of:\n";
9.      cout << "a: " << a << "\nb: " << b << "\nc: " << c << endl;
10.
11.             b = c * (b * (++a)--);  // b = 6 * ( 3 * 2) = 36
12.     cout << "Now, b = " << b << " and a = " << a << endl;
13.
14.             a = a >> b;           // a = 1 >> 36 = 0
15.     cout << "Final value of a = " << a << endl;
16.     return 0;
17. }
```

Standard Output

```
Initial values of:
a: 1
b: 3
c: 6
Now, b = 36 and a = 1
Final value of a = 0
```

Question Bank

Question 1.

Is subtraction(-) a unary operator or binary operator?

Subtraction operator is both unary and binary. In the statement -> int i=-3; Subtraction is a unary operator because the number of operands is one whereas in case of statement-> int i=6-3; subtraction is binary operator as the no of operators is two.

Question 2.

What are positive and negative numbers considered as, true or false?

Except 0 all the numbers (negative and positive) are treated as true.

E.g

```
if(-3)
{
Statement1;
}
Else{
Statement2;
}
```

Then statement1 will be executed.

Question3.

Give the output

```
#include <iostream>
using namespace std;
int main()
{
    int var1 = 9;
    int var2 = 6;
    if ((var2 = 2) == var1)
      cout << var2;
    else
       cout  << (var2 + 1);
}
```

The answer to the above code snippet is **3**. In if ((var2 = 2) == var1) we assign 2 to var2 variable hence the value of var2 is changed from 6 to 2 .Now since the expression inside if() evaluates to false the control is shifted to else part and the output is var2+1 which means 2+1 i.e 3.

Question 4.

Will the following code snippet give an error or not?

```
int a=83.2;

a=a%10;

cout<<a<<endl;
```

The code snippet will give a syntax error because the modulus operator(%) can't be used with float or double value.

Question 5.

What will be the output of the following statements?

1) **cout<<7/2<<endl;**
2) **cout<<7.5/2<<endl;**
3) **cout<<7.5/2.0<<endl;**
4) **cout<<7/2.0<<endl;**
5) **cout<<7.0/2.0<<endl;**

In the first statement both the operands are integers, so the output will be an integer.Therefore the output of the first statement is **3**.

In the second statement 7.5 if floating point type and 2 is integer type. So the output will be of floating point type. In this case the output will be **3.75**.

In the third statement 7.5 and 2.0 both are floating point type, therefore the result will be of floating point type. In this case output will be **3.75**.

In the fourth statement 2.0 is of floating point type, therefore the result will be of floating point type. In this case the output will be **3.5**.

In the fifth statement both 7.0 and 2.0 are of floating point type, therefore the result will be of floating point type. In this case the output will be **3.5**.

EXTRA QUESTION 1

1426A - Floor Number

https://codeforces.com/contest/1426/problem/A

This is a question from codeforces contest round 674 .

Let's see how to approach this problem.

In this question it is stated that Petya lives in a house which has various apartments on different floors.We have to find the floor on which Petya lives. We are given a number ' **n** ' which is the total number of

apartments in the whole house.It is given that the first floor has only 2 apartments. Every other floor has exactly '**x**' no of apartments.

Suppose take a case **n=7, x=3**

First floor will have 2 apartments as given in the question. Second floor has x=3 apartments . Therefore the 2nd floor has apartment_no3, apartment_no4, apartment_no5. Similarly the third floor will have 3 apartments: apartment_no6, apartment_no7, apartment_no8.
We know that the first floor contains only 2 apartments :apartment_no1 and apartment_no2. If we don't consider the first 2 apartments for the time being then the remaining (n-2) apartments are divided into floors of size "x".

- **If the (n-2) apartments are totally divisible by x then the total no of floors is 1+(n-2)/x**
  Suppose n=7 and x=5 so remaining apartments are (n-2) i.e 5. Now 5 is divisible by 5. Which means those remaining 5 apartments are on the same floor.
  [Case is same as, if we have 10rs and we want to buy pencils. Each pencil costs 2rs.Then we can buy only 10/2 pencils i.e 5 pencils.]
  Now the total no of apartments in this case would be 1+(5/5)=2
  [1 is added to (n-2)/x because we have to add the first floor too. Remember the first floor contains only 2 apartments.]
- **If the (n-2) apartments are not totally divisible by x then the total no of floors is 2+(n-2)/x**
  Suppose n=7 and x=2 so the remaining apartments are (n-2) i.e 5. 5 is not divisible by 2. Now 5%2=1 which means for 5/2 floors i.e for 2 floors we have 2 apartments each, but still 1 apartment is remaining which will come in the third floor.
  So total no of apartments in this case=1+(5/2)+1
  [Add first 1 for first floor and second 1 for any remaining apartment]

Link of code: https://sapphireengine.com/@/e26kyr
https://ideone.com/EocYtT

```cpp
#include<iostream>
using namespace std;
int main()
{

long long t;
cin>>t;
while(t--)
{

  long long n,x;
  cin>>n>>x;
  long long ans=0;
  if(n==2 || n==1)
  {
      cout<<1<<endl;
  }
  else if((n-2)%x==0)
  {
   ans=((n-2)/x)+1;
    cout<<ans<<endl;
  }
  else if((n-2)%x!=0)
  {
      ans=((n-2)/x)+2;
       cout<<ans<<endl;
  }

  }
  return 0;
  }
```

EXTRA QUESTION 2

[Problem - 4A]

https://codeforces.com/problemset/problem/4/A

This is a question from codeforces problem-set.

Let's see how to approach this problem.

In this question it is stated that two friends Pete and Billy bought a watermelon. Both of them are fond of even numbers so they want to divide the watermelon among themselves in such a way that both the parts are even numbers. It is not necessary that both parts should be equal.We have to say whether a watermelon of weight **"w"** can be divided into even parts or not.

Notice :

- **Even number + Even number= Even number**
- **Even number + Odd number= Odd number**

So we can say that if a number is odd it can't be represented as a sum of two even numbers. Every other number can be represented as a sum of two even numbers.

E.g:

10= 5+5 or 6+4

5=4+1 or 3+2

Also notice if the weight of the watermelon is 2 we can't divide it in 2 even parts.

So the answer is if the weight of the watermelon is odd or equal to 2 , it can't be divided in 2 even parts, otherwise it can be divided.

Program link: https://sapphireengine.com/@/3cl1x3
https://ideone.com/oeGbAZ

```
1. #include<iostream>
2. #include<bits/stdc++.h>
3. using namespace std;
4. int main()
5. {
6.  int w;
7.  cin>>w;
8.
```

```
9.  if(w%2!=0 || w==2)
10.    {
11.
12.        cout<<"NO"<<endl;
13.    }
14.   else{
15.      cout<<"YES"<<endl;
16.   }
17.     return 0;
18.
19.   }
```

NOTE: You can approach these questions differently also.

Practice questions:

1 **https://www.codechef.com/JAN20B/problems/BRKBKS**

2 Given n numbers, Raja is unable to find the number which is present odd number of times. If it's guaranteed that only one such number exists can you help Raja in finding the number which is present odd number of times?[Try to solve this using xor]

HINT: Truth table of xor is:

INPUTS		OUTPUTS
A	B	Y
0	0	0
0	1	1

1	0	1
1	1	0

NOTE: You don’t need to write the code for the questions as it requires the knowledge of loops. Just think about various approaches. In case of the first question think of an approach to build logic for a test case.

It’s ok if you can’t think of a logic for these questions, we will definitely learn more things in the coming lessons and will learn about different logics to solve the problems. The main motto of these questions is to make you familiar with operators.