Problem 1 Check1	Given two numbers x and y as input and print "true" if x >= y and "false" otherwise. Note: Don't use if-else to solve this problem.	
Test Cases	**Input**	**Output**
	x = 2, y = 3	false
	x = 10, y = 10	true
	x = 13, y = 9	true

Problem 2 Check2	Given two numbers a and b as input and print "true" if anyone or both of them are even. It will print "false" if both the numbers are odd. Note: Don't use if-else to solve this problem.	
Test Cases	**Input**	**Output**
	a = 7, b = 7	false
	a = 10, b = 13	true
	a = 31, b = 48	true

Problem 3 Multiple37	Given a number n as input and print "true" if it is divisible by either 3 or 7. Note: Don't use if-else to solve this problem.	
Test Cases	**Input**	**Output**
	n = 42	true
	n = 33	true
	n = 49	true
	n = 20	false

Problem 4 LargerThanOne	Given three numbers n1, n2 and num as input. Print "true" if num is greater than either n1 or n2. Note: Don't use if-else to solve this problem.	
Test Cases	**Input**	**Output**
	n1 = 12, n2 = 9, num = 15	true ⇨ because 15 is greater than both
	n1 = 11, n2 = 13, num = 12	true ⇨ because 12 is greater than 11
	n1 = 100, n2 = 25, num = 50	true
	n1 = 10, n2 = 20, num = 7	false

Problem 5 divide3not8	Given a number n as input and print "true" if n is is divisible by 3 but not divisible by 8. Note: Don't use if-else to solve this problem.	
Test Cases	**Input**	**Output**
	n = 15	true
	n = 24	false
	n = 7	false

TRISECT
www.trisectinstitute.com
9711848320

Problem 1 LeapYear	Given a number representing a year as input. Check if this year is a leap year or not. Print True or False accordingly. Leap Year: A year is a leap year if it is a multiple of 4 except century years. Century years (100, 200, 300 etc.) are only leap years if they are a multiple of 400. So 2000, 2400 are leap years because they are multiples of 400. 2100 and 2200 are not leap years because they are not multiples of 400.	
Test Cases	Input	Output
	1901	false
	2024	true
	1500	false
	1600	false

Problem 2 MaxOfThree	Given three numbers as input. Find the maximum number.	
Test Cases	Input	Output
	1, 5, 12	12
	6, 14, 2	14
	71, 8, 47	71
	-5, -2, -10	-2

Problem 3 AvgN	Given a number n as input. Print the average of 1 to n.	
Test Cases	Input	Output
	n = 2	1.5
	n = 5	7.5

Problem 4 GameFraction	Three friends A, B, C are playing a game. In this game an integer array is given. A has to speak fraction of all positive numbers except zero present in array, B has to speak fraction of all negative numbers and C has speak fraction of all zero present in array. This fractions should be in percentage. Print fraction spoken by A, B and C respectively in a array.	
Test Cases	Input	Output
	{4, 3, -9, 0, 4, 1}	{0.66, 0.16, 0.16} ⇨ {4/6, 1/6, 1/6}
	{1, 0, 0, -2, 4}	{0.40, 0.20, 0.40}
	{1, 13, -2, -4, 3}	{0.60, 0.40, 0}

Problem 5 CountVowel	Given a string as input count and print the number of vowels in the given string.	
Test Cases	**Input**	**Output**
	"read"	2
	"xmas"	1

Problem 6 CaesarCipher	Given a string and integer n as input. It then prints a new string in which each character is shifted n times (think circular). Assume all characters are small. For example if n = 2 then "abz" will become "cdb" 'a' + 2 => 'c', 'b' + 2 => d Important: (Circular Shifting) 'z' + 2 => 'b' Note: If the resultant value after shifting is outside the range of alphabets then we can use modulo operation to bring it inside the range i.e. base + (((character + key) – base) % totalCharacters) Here, Let Input: ch and n base = 'a' (= 97) totalChars = 26 shift = ch + n circularShift = (shift – base) % totalChars newChar = (char) (base + circularShift) Solve For: ch = 'x' and n = 5 shift = 'x' + 5 => 125 circularShift = (125 – 97) % 26 => 28%26 => 2 newChar = (char) (97 + 2) => (char) (99) => 'c'	
Test Cases	**Input**	**Output**
	"read", 1	"sfbe"
	"xmas", 4	'bqew'

Problem 1 CharDistance	Given a character as input and print it's distance from character 'D'.	
Test Cases	**Input**	**Output**
	`charDistance('A')`	`-3`
	`charDistance('d')`	`32`
	`charDistance('F')`	`2`

Problem 2 ToUpperCase	Given a character in lowercase as input and print its corresponding uppercase character. Note: There is a difference of 32 characters between a character and its opposite case.	
Test Cases	**Input**	**Output**
	`toUpperCase('a')`	`'A'`
	`toUpperCase('d')`	`'D'`

Problem 3 ToLowerCase	Given a character in uppercase as input and print its corresponding lowercase character. Note: There is a difference of 32 characters between a character and its opposite case.	
Test Cases	**Input**	**Output**
	`toLowerCase('I')`	`'i'`
	`toLowerCase('H')`	`'h'`

New Data Type: Char

We have already used character data type while working with Strings. The programs however didn't perform any operations on char variables.

Example 1

```
1   char ch = 'a';
2   System.out.println(ch);
3   ch = 'A';
4   System.out.println(ch);
5   ch = '#';
6   System.out.println(ch);
7   ch = '5';
8   System.out.println(ch);
```

Output:

```
a
A
#
5
```

Explanation:

Character variables can hold any valid character value – alphabets, digits or any special character. Additionally, alphabets can be either be in lowercase or in uppercase.

Character to Integer Conversion

Character values can be converted into their corresponding ASCII values (integer).

Example 2

```
1   char ch = 'A';
2   System.out.println(ch);
3   int num = ch;                 //notice: data type is int, automatic conversion
4   System.out.println(num);
```

Output:

```
A
65
```

Explanation:

ASCII value of 'A' is 65.

Important ASCII ranges:

Expression		ASCII Range
Uppercase Alphabets	['A' – 'Z']	65 – 90
Lowercase Alphabets	['a' – 'z']	97 – 122
Digits	['0' – '9']	48 – 57

Example 3

```
1   int ch = 'C';
2   int num = ch;                          //uppercase alphabets start from 65
3   System.out.println(num);
4   num = 'd';                             //lowercase alphabets start from 97
5   System.out.println(num);
6   num = 8;                               //integer value
7   System.out.println(num);
8   num = '5';                             //digit
9   System.out.println(num);
```

Output:

```
67
100
8
53
```

Problem 1	**Give output of following code.**
``1 char x = 'a'; 2 char y = 'A'; 3 boolean bool = (x == y); 4 System.out.println(bool);```	
Output When: x = 'a', y = 'A' x = 'b', y = 'b'	

Problem 2	**Give output of following code.**
``1 char x = 'a'; 2 char y = 'A'; 3 boolean bool = (x > y); 4 System.out.println(bool);```	
Output When: x = 'a', y = 'A' x = 'd', y = 'd'	

Problem 3	**Give output of following code.**
``1 int num = 3; 2 char multiple = 'A'; 3 if(multiple >= '1' && multiple <= '6') 4 { 5 num = num * 2; 6 } 7 else 8 { 9 num = num * 10; 10 } 11 System.out.println(num);```	

Output When:

```
num = 3, multiple = 'A'
num = 6, multiple = '2'
```

Integer to Character

For conversion from integer to char we need to explicitly mention the data type otherwise we'll get a compile time error.

Example 4

```
1   int num = 67;
2   char ch = (char) num;
3   System.out.println(ch);
4   ch = (char) (num + 2);
```

Output:

```
C
E
```

Character Arithmetic

We can perform arithmetic operations on characters just like we do with integers. The result of arithmetic operation in characters is always in integer. So if you want to store it in character again then you need to explicitly cast the result.

Example 5

```
1   char ch = 'D';
2   int num = ch + 1;                   //result in integers
3   System.out.println(num);
4   char ch2 = (char) (ch - 1);         //necessary to mention data type
```

Output:

```
69
C
```

Loss of Precision Error:

"Loss of precision" error is generated whenever we try to store integer variable into character without explicitly mentioning the conversion.

Example 10.1:

```
int n = 65;
char ch = n;              //compile time error: loss of precision
```

Example 10.2:

```
int n = 65;
char ch = (char) n;
System.out.println(ch);   //gives A
```

Character to Integer	Integer to Character
`char ch = 'a';` **Code 1: Character variable to integer** `int x = ch;` `System.out.println(x);` **Code 2: Explicitly mentioning the conversion** `int x = (int) ch;` `System.out.println(x);` **Code 3: Character arithmetic** `int x = 'd' + 2;` `System.out.println(x);`	`int n = 65;` `char ch1 = 'd';` **Code 4: Integer to character** `char ch2 = n;` `System.out.println(ch2);` **Code 5: Explicitly mentioning the conversion** `char ch2 = (char) n;` `System.out.println(ch2);` **Code 6: Integer expression** `char ch2 = (char) (n + 1);` `System.out.println(ch2);` **Code 7: Character arithmetic** `char ch2 = (char) (ch1 + 2);` `System.out.println(ch2);`
Output: **Code 1:** 97 **Code 2:** 97 **Code 3:** 102	**Output:** **Code 4:** Error: loss of precision **Code 5:** A **Code 6:** B **Code 7:** f

Problem 4	**Give value of variable x for each of the following statements.**

```
1) 2 + 'b'                : _______
2) '2' - 2                : _______
3) '2' + '2'              : _______
4) ((char) 65 ) + 3       : _______
5) (char) (97 + 1)        : _______
```

Problem 5	Give output of following code.

```
1   char x = 'a'
2   char y = (char)(x + 10);
3   char z = (char)(x + 'A' - 'a');
4   System.out.println(z);
5   System.out.println(y);
```

Output When:

```
x = 'a'
x = 'c'
```

Problem 6	Give output of following code.

```
1   int num = 6;
2   char multiple = 'a';
3   int diff = multiple - '0';
4   if ((diff >= 0) && (diff <= 6))
5   {
6      num = diff * num;
7   }
8   num = 10 * num;
9   System.out.println(num);
```

Output When:

```
num = 6 multiple = 'a'
num = 7 multiple = '2'
num = 8 multiple = '1'
```

Problem 1 Average	Given four integer values as input and print average of these numbers.	
Test Cases	**Input**	**Output**
	`average(1, 2, 4, 1)`	`4.0`
	`average(2, 3, 4, 1)`	`2.5`

Problem 2 Percentage	Given marks of a student in three subjects as input, print the overall percentage. Note: maximum marks in each subject is 100.	
Test Cases	**Input**	**Output**
	`percentage(80, 90, 70)`	`80.0`
	`percentage(60, 75, 85)`	`73.33`

Problem 3 Invoice	Given item quantity, price and tax rate as input. Print the total tax to be paid (without the decimal part).	
Test Cases	**Input**	**Output**
	`invoice(100, 2, 10.1)`	`20`
	`Invoice(5, 2, 3.5)`	`0`

New Data Type: Double

So far we have used int data type while creating our variables containing numbers. However, int can only hold integer values. When it comes to decimal values we cannot use int. To hold decimal values we have special data type called double.

Example 1

```
1   double decimalValue = 10.5;                //notice: decimal value
2   System.out.println(decimalValue);
```

On running above code you'll see following output.

Output:

```
10.5
```

Note: However if your try to replace double with int in above code you'll get compiler error.

Example 2

```
1   int decimalValue = 10.5;                   //notice: data type is int
2   System.out.println(decimalValue);
```

On running above code you'll see following error.

Output:

```
Line 1: error: possible loss of precision
             int decimalValue = 10.5;
```

So, if you want to store decimal values then always use double data type.

Problem 1	**Give output of following code.**

```
1   double length = 10.2;
2   double breadth = 10.4;
3   double height = 5.2;
4   double volume = length * breadth * height;
5   System.out.println(volume);
```

Problem 2	**Give output of following code.**

```
1   double amount = 1200.0;
2   double discount = 25.0;       //in percent
3   double amountPaid = 0.0;
4   double change = 0.0;
5   amount = amount - ((amount * discount) / 100.0);
6   amountPaid = 1000;
7   change = amountPaid - amount;
8   System.out.println("Pay Change: " + change);
```

Special Case: Storing integer values in double variables

So far we stored decimal values in double variables. But, we can store integers in double also. They get automatically converted into equivalent double value.

Example 3

```
1   int i = 20;
2   double d = 10;
3   System.out.println(d);
4   d = i;
5   System.out.println(d);
```

Output [Notice how instead of 10 we get 10.0 in output]:

```
10.0
20.0                              //20 automatically gets converted into 20.0
```

Type Casting: Conversion between data types

Type Casting is the mechanism of converting a value from one data type to another. However, keep in mind only compatible data types can be converted to each other.

Integer to Double

Example 4

```
1   int div1 = 10/4;                  //notice: data type is int
2   System.out.println(div1);
3   double div2 = 10/4;               //notice: data type is double
4   System.out.println(div2);
```

Output:

```
2
2.0
```

Explanation:

Expression: 10/4

Given expression contains both integer operands. It is of the form integer/integer. This always generate answer in integer which is 2.

Line 1: We are storing answer of our expression (which is 2) in integer variable hence div1 = 2.

Line 3: We are storing answer of our expression in double variable. It automatically gets converted into equivalent double value (which is 2.0). This is called implicit type casting.

To get answer of our division in decimal we should convert any our operand to double. Consider following table:

Expression	Output	Example	Answer
integer/integer	integer	10/4	2
integer/double	double	10/4.0	2.5
double/integer	double	10.0/4	2.5

double/double	double	10.0/4.0	2.5

In fact any arithmetic operation with a double value will generate double value. So the final answer will always be in double.

Example 5

```
2 + 4 = 6
2 + 4.0 = 6.0
3 * 2.1 = 6.3
```

Example 6

```
1   int x = 10;
2   int y = 20;
3   y = y/x;
4   System.out.println(y);
5   y = x + x/y;
6   System.out.println(y);
```

Output:

```
2
15
```

Example 7

```
1   int x = 9;
2   int y = 4;
3   double z = x/y;
4   System.out.println(z);
5   z = x/4.0;
6   System.out.println(z);
```

Line by line execution of Example 7

Line	Operation	Calculation	Variables			Screen
			x	y	z	
1	New Value	x = 9	9	-	-	-
2	New Value	y = 4	9	4	-	-
3	New Value	z = 9/4 but, 9/4 = 2 [int/int] z = 2.0	9	4	2.0	-
4	Print	-	9	4	2.0	2.0
5	New Value	z = 9/4.0 [int/double] z = 2.25	9	4	2.25	-
6	Print	-	9	4	2.25	2.25

Problem 3	**Give output of following code.**

```
1   int a = 10;
2   double b = 4;
3   int c = 4;
4   System.out.println(b);
```

```
5   b = a/c;
6   System.out.println(b);
7   b = 10/b;
8   System.out.println(b);
```

Problem 4	Solve following expressions.

```
1) 10 / 2                : _______
2) 10.0 / 5              : _______
3) 4 + (2 / 4)           : _______
4) (2.0 / 4) + 3         : _______
5) 1 / (2 * 1.0)         : _______
6) 10 / (16 / 4)         : _______
```

Explicitly stating data type of conversion

So far we have converted direct integer values to decimal values. However, we cannot apply the same mechanism when we want to convert integer variables into double values. For this we need to explicitly state double while converting otherwise we'll might not get the desired answer.

Syntax:

```
variable = (data-type) expression;
```

Example 8

```
1   int x = 10;
2   int y = 4;
3   double d = x/y;
4   System.out.println(d);
5   d = (double) (x/y);                  //working with variables: explicit
    conversion
6   System.out.println(d);
7   d = ((double) x)/y;                  //explicit stating conversion
8   System.out.println(d);
```

Line by line execution of Example 8

Line	Operation	Calculation	Variables			Screen
			x	y	d	
1	New Value	x = 10	10	-	-	-
2	New Value	y = 4	10	4	-	-
3	New Value	d = 10/4 but, 10/4 = 2 [integer/integer division] d = 2.0	10	4	2.0	-
4	Print	-	10	4	2.0	2.0

5	New Value	d = (double) (x/y) d = (double) (10/4) d = (double) 2 d = 2.0	10	4	2.0	2.0
6	Print	-	10	4	2.0	2.0
7	New Value	d = ((double) x)/y d = ((double) 10)/4 d = (10.0)/4 [(double 10) = (10.0)] d = 2.5 [double/integer division]	10	4	2.5	2.5
8	Print	-	10	4	2.5	2.5

Double to Integer

While converting double to integer it is mandatory to explicitly mention the data type of conversion otherwise we'll get a compile time error.

Example 9

```
1  double d = 10.4;
2  int x = (int) d;          //converting double to int (decimal part gets
   truncated)
3  System.out.println(x);
4  x = (int) 10.0/4.0;
5  System.out.println(x);
```

Line by line execution of Example 9

Line	Operation	Calculation	Variables		Screen
			d	x	
1	New Value	10.4	10.4	-	-
2	New Value	x = (int) d x = (int) 10.4 x = 10	10.4	10	-
3	Print	-	10.4	10	10
4	New Value	x = (int) 10.0/4.0 x = (int) 2.5 [double/double] x = 2	10.4	2	-
5	Print	-	10.4	2	2

Loss of Precision Error:

"Loss of precision" error is generated whenever we try to store decimal value (double values) into integers. To overcome this we need to explicitly mention that we want to convert into integer which will result in loss of information.

Example 10.1:

```
int x = 10.5;              //compile time error: loss of precision
```

Example 10.2:

```
int y = (int) 10.5;        //we lose information after decimal
System.out.println(y);     //gives 10
```

Integer to Double	Double to Integer
`int a = 10;` **Code 1: Integer value to double** `double b = 10;` `System.out.println(b);` **Code 2: Integer variable to double** `double b = a;` `System.out.println(b);` **Code 3: Mentioning conversion explicitly** `double b = (double) a;` `System.out.println(b);` **Code 4: Integer / Integer** `double b = a / 4;` `System.out.println(b);` **Code 5: Double (explicit) / Integer** `double b = ((double) a) / 4;` `System.out.println(b);` **Code 6: Integer / Double Value** `double b = a / 4.0;` `System.out.println(b);`	`int x = 20, y = 10;` **Code 7: Double value to integer** `int z = 10.0;` `System.out.println(z);` **Code 8: Integer variable converted to double** `int z = (double) x;` `System.out.println(z);` **Code 9: Integer / Integer** `int z = x / y;` `System.out.println(z);` **Code 10: Integer / Double** `int z = x / 5.0;` `System.out.println(z);` **Code 11: Converting result into integer** `int z = (int) (y / 4.0);` `System.out.println(z);` **Code 12: Double value to integer** `int z = x + 2.5;` `System.out.println(z);`
Output: **Code 1:** 10.0 **Code 2:** 10.0 **Code 3:** 10.0 **Code 4:** 2.0 **Code 5:** 2.5 **Code 6:** 2.5	**Output:** **Code 7:** Error: loss of precision **Code 8:** Error: loss of precision **Code 9:** 2 **Code 10:** Error: loss of precision **Code 11:** 2 **Code 12:** Error: loss of precision

Problem 5	Give output of following code.

```
1   int seconds = 1054800;
2   double hours = seconds/3600.0;
3   System.out.println("Total hours of Work: "+hours);
4   int days = (int) hours/24;
5   System.out.println("Total days of Work: "+days);
```

Problem 6	Give value of variable x for each of the following statements.

```
1) int x = 10 + (5 / 2);                        : _______
2) int x = 5 / 2.0;                             : _______
3) int x = 5 + 2.5;                             : _______
4) int x = (int) (10 + 3.2);                    : _______
5) int x = (double) (10 + 2);                   : _______
6) int y = 2;
   int x = (int) (7 / ((double) y) + 3.5);      : _______
7) int x = (int) (7 / y + 3.5);                 : _______
```

TRISECT
www.trisectinstitute.com
9711848320

Boolean Data Type

Java provides a dedicated data type to hold values of expressions that evaluates to either true or false.

Note: boolean data types can only hold either true or false values.

Syntax:

```
boolean variable_name = value or expression;
```

Example 1:

```
1  boolean bool = true;
2  System.out.println(bool);          //true
3
4  bool = 2 > 4;                      //false
5  System.out.println(bool);
6
7  bool = 6 <= 7;                     //true
8  System.out.println(bool);
9
10 int x = 7%2;
11 bool = (x == 0);                   //false
12 System.out.println(bool);
```

Output:

```
true
false
true
false
```

Boolean Data Type: Basic Operations

You can use any logical operator (AND, OR , NOT) with boolean variables/values.

Quick Sheet			
bool1	**bool2**	**bool1 && bool2**	**bool1 || bool2**
true	false	false	true
false	true	false	true
false	false	false	false
true	true	true	true

true	false
false	true

Truth Table NOT (!)	
Boolean value 1	**Result**

Example 2:

```
1  boolean bool1 = true;
2  boolean bool2 = false;
3
4  boolean bool3 = bool1 && bool2;
5  System.out.println(bool3);
6
7  bool3 = bool1 || bool2;
8  System.out.println(bool3);
9
10 bool3 = !bool1;
11 System.out.println(bool3);
12
13 bool3 = bool1 && (!bool2);
14 System.out.println(bool3);
```

Output:

```
false
true
false
true
```

Problem 1	**Give output of following code.**

```
1  boolean bool1 = true;
2  boolean bool2 = false;
3  boolean bool3 = true;
4  boolean bool4 = bool1 && bool2;
5  boolean bool = bool4 && bool3;
6  System.out.println(bool);
7  System.out.println(bool4);
```

Problem 2	**Give output of following code.**

```
1  boolean bool1 = true;
2  boolean bool2 = false;
3  bool1 = !bool1;
4  bool2 = !bool2;
5  boolean boolA = (bool1 || bool2);
6  boolean boolB = (bool1 && bool2);
7  boolean bool = boolA && boolB;
8  System.out.println(bool);
9  System.out.println(boolA);
10 System.out.println(boolB);
```

Example 3: Given two integers as input and print "true" if both of them are even numbers. Otherwise print "false".

```
1   int x = 4;
2   int y = 22;
3   boolean bool1 = x % 2 == 0;
4   boolean bool2 = y % 2 == 0;
5   boolean bool3 = bool1 && bool2;
6   System.out.println(bool3);
```

Problem 3	**Refer the code given below and give output for input values given after.**

```
1   int a = 57;
2   int b = 40;
3   boolean b1 = (a + b) > 100;
4   boolean b2 = a % 2 == 0;
5   boolean b3 = b1 && b2;
6   boolean b4 = b1 || b2;
7   boolean bool = b3 || b4;
8   System.out.println(bool);
```

Output when:

```
a = 57, b = 40
a = 113, b = 60
a = 32, b = 70
```