NAME:ROHAN NYATI

SAP ID:500075940

ROLL NO. : R177219148

BATCH-5(AI&ML)

Experiment

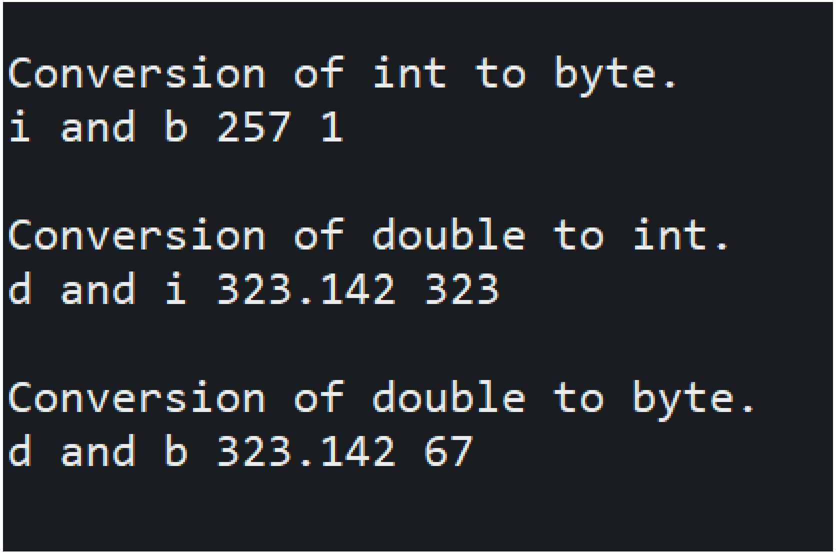
TITLE: Statements & Blocks, if-else, switch, while and do-while, for, Labels, break, continue, return, goto

1) Casting Incompatible Types

Code ->

```
// Demonstrate casts.
public class Conversion {

    public static void main(String[] args) {
        byte b;
        int i = 257;
        double d = 323.142;

        System.out.println("\nConversion of int to
byte.");
        b= (byte) i;
        System.out.println("i and b " + i + " " + b);

        System.out.println("\nConversion of double to
int.");
        i = (int) d;
        System.out.println("d and i " + d + " " + i);

        System.out.println("\nConversion of double to
byte.");
        b = (byte) d;
        System.out.println("d and b " + d + " " + b);
    }
}
```

Output ->

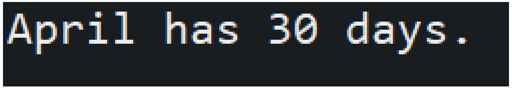


```
Conversion of int to byte.
i and b 257 1

Conversion of double to int.
d and i 323.142 323

Conversion of double to byte.
d and b 323.142 67
```

2) Array Initialization
Code ->

```
// Demonstrate a one-dimensional array.
public class Array {

    public static void main(String[] args) {
        int month_days [];
        month_days = new int [12];
        month_days [0] = 31;
        month_days [1] = 28;
        month_days [2] = 31;
        month_days [3] = 30;
        month_days [4] = 31;
        month_days [5] = 30;
        month_days [6] = 31;
        month_days [7] = 31;
        month_days [8] = 30;
        month_days [9] = 31;
        month_days [10] = 30;
        month_days [11] = 31;

        System.out.println("April has " + month_days [3]
+ " days.");
    }
}
```

Output ->



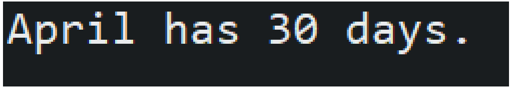
```
April has 30 days.
```

3) Improved version of the above code

Code ->

```
// An improved version of the previous program.
public class AutoArray {

     public static void main(String[] args) {

          int month_days [] = { 31, 28, 31, 30, 31, 30, 31,
31, 30, 31, 30, 31 };

          System.out.println("April has " + month_days [3]
+ " days.");
     }
}
```

Output ->



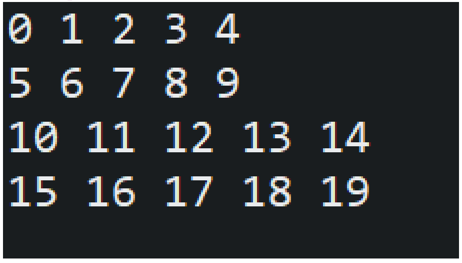
```
April has 30 days.
```

4) Multi-Dimensional Arrays

Code ->

```
// Demonstrate a two-dimensional array.
public class TwoDArray {

     public static void main(String[] args) {
          int twoD [] [] = new int [4] [5];
          int i, j, k = 0;

          for (i=0; i<4; i++)
               for (j=0; j<5; j++)
               {
                    twoD [i] [j] = k;
                    k++;
               }
          for (i=0; i<4; i++)
          {
            for(j=0; j<5; j++)
                System.out.print(twoD [i] [j] + " ");
            System.out.println();
        }
    }
}
```

Output ->



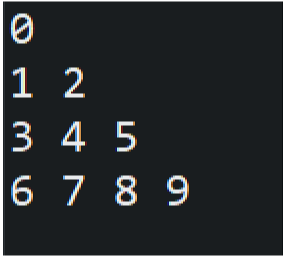
```
0 1 2 3 4
5 6 7 8 9
10 11 12 13 14
15 16 17 18 19
```

5) Manually allocating size of 2-D

Code ->

```
// Manually allocate differing size second dimensions.
public class TwoDAgain {

    public static void main(String[] args) {
        int twoD [] [] = new int [4] [];
        twoD [0] = new int [1];
        twoD [1] = new int [2];
        twoD [2] = new int [3];
        twoD [3] = new int [4];

        int i, j, k = 0;

        for (i=0; i<4; i++)
            for(j=0; j<i+1; j++)
            {
                twoD[i] [j] = k;
                k++;
            }

        for (i=0; i<4; i++)
        {
            for (j=0; j<i+1; j++)
                System.out.print(twoD [i] [j] + " ");
            System.out.println();
        }
    }
}
```

Output ->



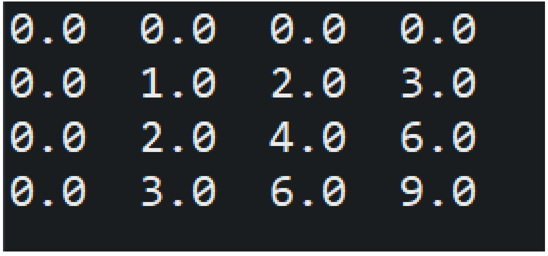
```
0
1 2
3 4 5
6 7 8 9
```

6) Initializing 2-D Array

Code ->

```
//Initialize a two-dimensional array.
public class Matrix {

    public static void main(String[] args) {
        double m[] [] = {
                { 0*0, 1*0, 2*0, 3*0 },
                { 0*1, 1*1, 2*1, 3*1 },
                { 0*2, 1*2, 2*2, 3*2 },
                { 0*3, 1*3, 2*3, 3*3 }
        };
        int i, j;

        for (i=0; i<4; i++)
        {
            for(j=0; j<4; j++)
                System.out.print(m [i] [j] + "  ");
            System.out.println();
        }
    }
}
```

Output ->

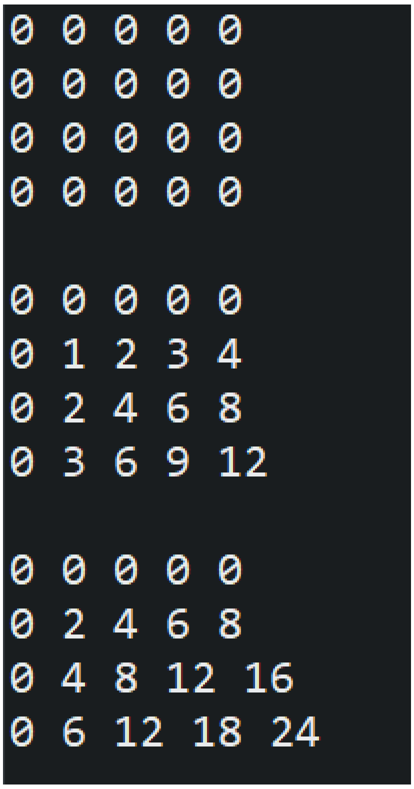


```
0.0   0.0   0.0   0.0
0.0   1.0   2.0   3.0
0.0   2.0   4.0   6.0
0.0   3.0   6.0   9.0
```

7) Demonstrating a 3-D Array
Code ->

```
// Demonstrate a three-dimensional array.
public class ThreeDMatrix {

    public static void main(String[] args) {
        int threeD [] [] [] = new int [3] [4] [5];
        int i, j, k;

        for (i=0; i<3; i++)
            for(j=0; j<4; j++)
                for(k=0; k<5; k++)
                    threeD [i] [j] [k] = i * j * k;

        for (i=0; i<3; i++)
        {
            for (j=0; j<4; j++)
            {
                for (k=0; k<5; k++)
                    System.out.print(threeD [i] [j] [k]
+ " ");
                System.out.println();
            }
            System.out.println();
        }
    }
}
```

Output ->



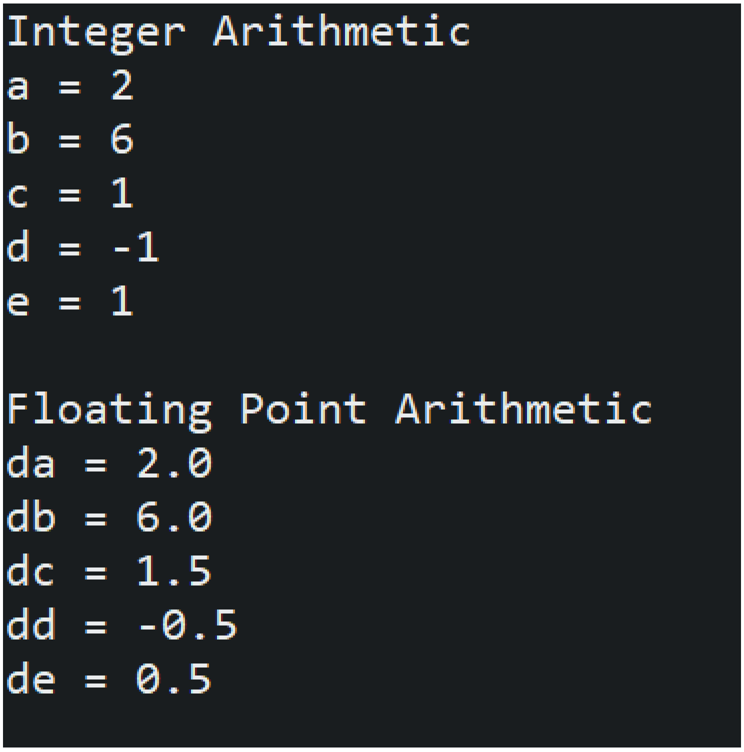
```
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0

0 0 0 0 0
0 1 2 3 4
0 2 4 6 8
0 3 6 9 12

0 0 0 0 0
0 2 4 6 8
0 4 8 12 16
0 6 12 18 24
```

8) The basic arithmetic operators
Code ->

```
// Demonstrate the basic arithmetic operators.
public class BasicMath {

	public static void main(String[] args) {
		// arithmetic using integers
		System.out.println("Integer Arithmetic");
		int a = 1 + 1;
		int b = a * 3;
		int c = b / 4;
		int d = c - a;
		int e = -d;
		System.out.println("a = " + a);
		System.out.println("b = " + b);
		System.out.println("c = " + c);
		System.out.println("d = " + d);
		System.out.println("e = " + e);

		// arithmetic using doubles
		System.out.println("\nFloating Point 
Arithmetic");
```

```
        double da = 1 + 1;
        double db = da * 3;
        double dc = db / 4;
        double dd = dc - a;
        double de = -dd;
        System.out.println("da = " + da);
        System.out.println("db = " + db);
        System.out.println("dc = " + dc);
        System.out.println("dd = " + dd);
        System.out.println("de = " + de);
    }
}
```

Output ->

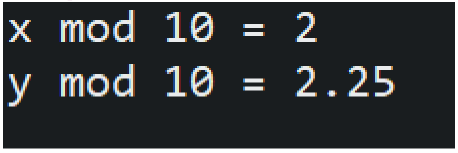


```
Integer Arithmetic
a = 2
b = 6
c = 1
d = -1
e = 1

Floating Point Arithmetic
da = 2.0
db = 6.0
dc = 1.5
dd = -0.5
de = 0.5
```

9) The Modulus operator
Code ->

```
// Demonstrate the % operator.
public class Modulus {

    public static void main(String[] args) {
        int x = 42;
        double y = 42.25;
        System.out.println("x mod 10 = " + x % 10);
        System.out.println("y mod 10 = " + y % 10);
    }
}
```

Output ->



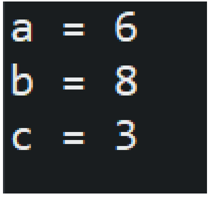
```
x mod 10 = 2
y mod 10 = 2.25
```

10) Arithmetic Compound Assignment operators

Code ->

```
// Demonstrate several assignment operators.
public class OpEquals {

    public static void main(String[] args) {
        int a = 1;
        int b = 2;
        int c = 3;

        a += 5;
        b *= 4;
        c += a * b;
        c %= 6;
        System.out.println("a = " + a);
        System.out.println("b = " + b);
        System.out.println("c = " + c);
    }
}
```

Output ->



```
a = 6
b = 8
c = 3
```

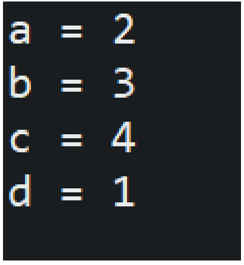
11) Increment and Decrement

Code ->

```
// Demonstrate ++.
```

```
public class IncDec {

    public static void main(String[] args) {
        int a = 1;
        int b = 2;
        int c;
        int d;
        c = ++b;
        d = a++;
        c++;
        System.out.println("a = " + a);
        System.out.println("b = " + b);
        System.out.println("c = " + c);
        System.out.println("d = " + d);
    }
}
```

Output ->



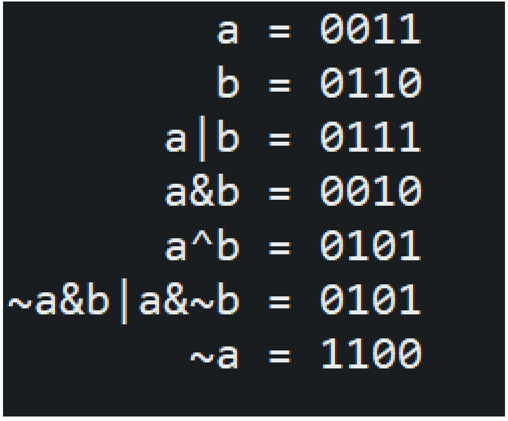
```
a = 2
b = 3
c = 4
d = 1
```

12) The Bitwise Logical Operators

Code ->

```
public class BitLogic {

    public static void main(String[] args) {
        String binary [] = {
                "0000", "0001", "0010", "0011", "0100",
"0101", "0110", "0111", "1000", "1001", "1010", "1011",
"1100", "1101", "1110", "1111"
        };
        int a = 3; // 0 + 2 + 1 or 0011 in binary
        int b = 6; // 4 + 2 + 0 or 0110 in binary
        int c = a | b;
        int d = a & b;
        int e = a ^ b;
        int f = (~a & b) | (a & ~b);
        int g = ~a & 0x0f;

        System.out.println("        a = " + binary [a]);
        System.out.println("        b = " + binary [b]);
        System.out.println("      a|b = " + binary [c]);
        System.out.println("      a&b = " + binary [d]);
        System.out.println("      a^b = " + binary [e]);
        System.out.println("~a&b|a&~b = " + binary [f]);
        System.out.println("       ~a = " + binary [g]);
    }
}
```

Output ->

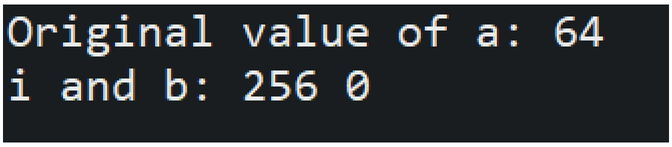


```
        a = 0011
        b = 0110
      a|b = 0111
      a&b = 0010
      a^b = 0101
~a&b|a&~b = 0101
       ~a = 1100
```

13) The Left Shift
Code ->

```
// Left shifting a byte value.
public class ByteShift {

    public static void main(String[] args) {
        byte a = 64, b;
        int i;

    i = a << 2;
    b = (byte) (a << 2);

    System.out.println("Original value of a: " + a);
    System.out.println("i and b: " + i + " " + b);
    }
}
```

Output ->

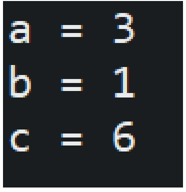


```
Original value of a: 64
i and b: 256 0
```

14) Bitwise operator compound assignments
Code ->

```
public class OpBitEquals {

	public static void main(String[] args) {
		int a = 1;
		int b = 2;
		int c = 3;

		a |= 4;
		b >>= 1;

		c <<= 1;
		a ^= c;
		System.out.println("a = " + a);
		System.out.println("b = " + b);
		System.out.println("c = " + c);
	}
}
```

Output ->



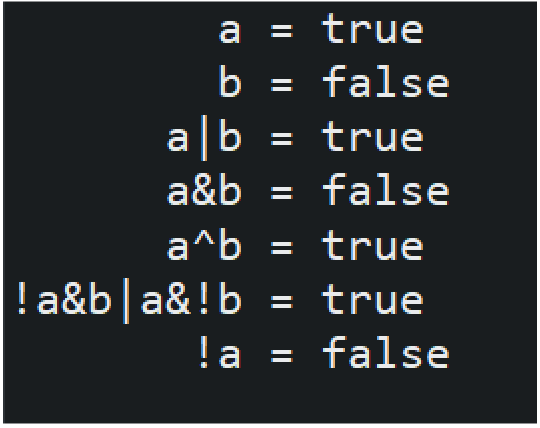
```
a = 3
b = 1
c = 6
```

15) Boolean Logic operators
Code ->

```
// Demonstrate the boolean logical operators.
public class BoolLogic {

	public static void main(String[] args) {
		boolean a = true;
		boolean b = false;
		boolean c = a | b;
		boolean d = a & b;
```

```
        boolean e = a ^ b;
        boolean f = (!a & b) | (a & !b);
        boolean g = !a;

        System.out.println("        a = " + a);
        System.out.println("        b = " + b);
        System.out.println("      a|b = " + c);
        System.out.println("      a&b = " + d);
        System.out.println("      a^b = " + e);
        System.out.println("!a&b|a&!b = " + f);
        System.out.println("       !a = " + g);
    }
}
```

Output ->



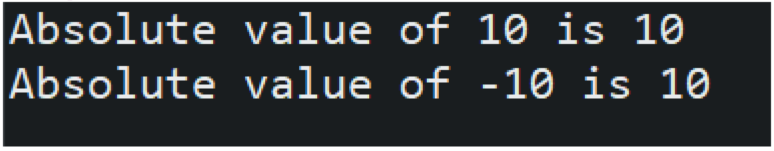
```
        a = true
        b = false
      a|b = true
      a&b = false
      a^b = true
!a&b|a&!b = true
       !a = false
```

16) The Assignment operator
Code ->

```
// Demonstrate ?.
public class Ternary {

    public static void main(String[] args) {
        int i, k;

        i = 10;
        k = i < 0 ? -i : i; // get absolute value of i
        System.out.print("Absolute value of ");
        System.out.println(i + " is " + k);

        i = -10;
        k = i < 0 ? -i : i; // get absolute value of i
```

```
            System.out.print("Absolute value of ");
            System.out.println(i + " is " + k);
        }
}
```

Output ->



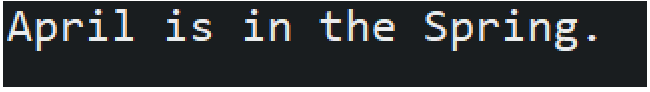
```
Absolute value of 10 is 10
Absolute value of -10 is 10
```

17) The if-else-if ladder

Code ->

```
// Demonstrate if-else-if statements.
public class IfElse {

        public static void main(String[] args) {
            int month = 4; // April
            String season;

            if (month == 12 || month == 1 || month == 2)
                season = "Winter";
            else if (month == 3 || month == 4 || month == 5)
                season = "Spring";
            else if (month == 6 || month == 7 || month == 8)
                season = "Summer";
            else if (month == 9 || month == 10 || month ==
11)
                season = "Autumn";
            else
                season = "Bogus Month";

            System.out.println("April is in the " + season +
".");
        }
}
```

Output ->



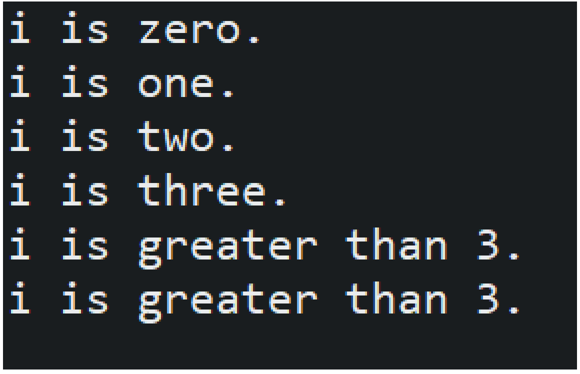
```
April is in the Spring.
```

18) Switch Case

Code ->

```
// A simple example of the switch.
public class SampleSwitch {

	public static void main(String[] args) {
		for (int i=0; i<6; i++)
			switch (i)
			{
			case 0:
				System.out.println("i is zero.");
				break;
			case 1:
				System.out.println("i is one.");
				break;
			case 2:
				System.out.println("i is two.");
				break;
			case 3:
				System.out.println("i is three.");
				break;
			default:
				System.out.println("i is greater than 
3.");
			}
	}
}
```

Output ->



```
i is zero.
i is one.
i is two.
i is three.
i is greater than 3.
i is greater than 3.
```

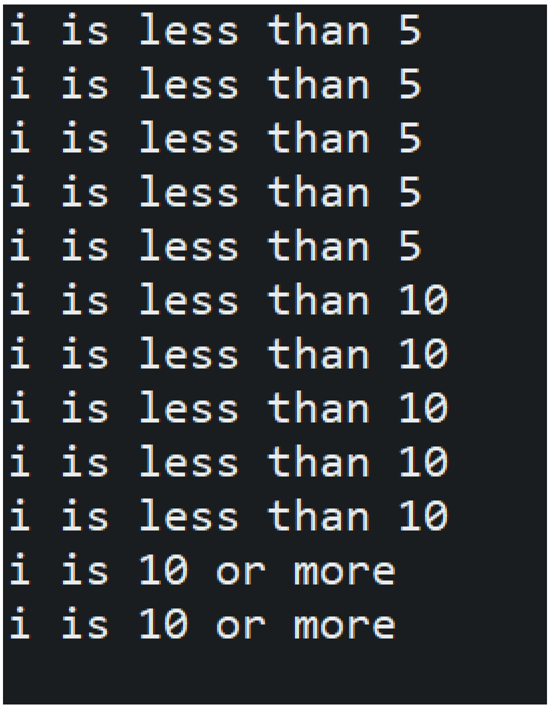
19) Switch case with missing break statement

Code ->

```
// In a switch, break statements are optional.
```

```
public class MissingBreak {

    public static void main(String[] args) {
        for (int i=0; i<12; i++)
            switch (i)
            {
            case 0:
            case 1:
            case 2:
            case 3:
            case 4:
                System.out.println("i is less than 5");
                break;
            case 5:
            case 6:
            case 7:
            case 8:
            case 9:
                System.out.println("i is less than
10");
                break;
            default:
                System.out.println("i is 10 or more");
            }
    }
}
```

Output ->



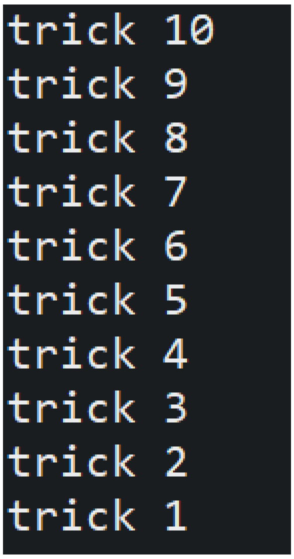
```
i is less than 5
i is less than 5
i is less than 5
i is less than 5
i is less than 5
i is less than 10
i is less than 10
i is less than 10
i is less than 10
i is less than 10
i is 10 or more
i is 10 or more
```

20) Demonstrate the while loop

Code ->

```
// Demonstrate the while loop.
public class While {

	public static void main(String[] args) {
		int n = 10;

		while (n > 0)
		{
			System.out.println("trick " + n);
			n--;
		}
	}
}
```

Output ->

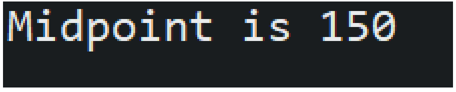


```
trick 10
trick 9
trick 8
trick 7
trick 6
trick 5
trick 4
trick 3
trick 2
trick 1
```

21) The target of a loop can be empty
Code ->

```
// The target of a loop can be empty.
public class NoBody {

	public static void main(String[] args) {
		int i, j;

		i = 100;
		j = 200;

		// find midpoint between i and j
		while (++i < --j); // no body in this loop

		System.out.println("Midpoint is " + i);
	}
}
```

Output ->

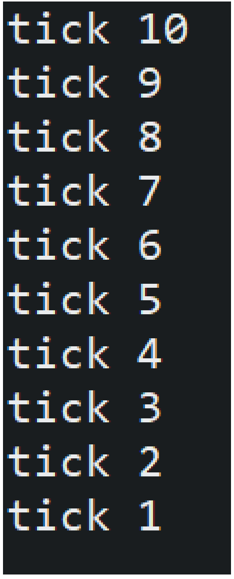


```
Midpoint is 150
```

22) Do-While
Code ->

```
// Demonstrate the do-while loop.
public class DoWhile {
    public static void main(String[] args) {
        int n = 10;

        do {
            System.out.println("tick " + n);
            n--;
        } while (n > 0);
    }
}
```

Output ->



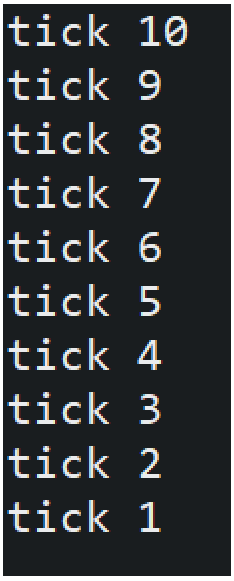
```
tick 10
tick 9
tick 8
tick 7
tick 6
tick 5
tick 4
tick 3
tick 2
tick 1
```

23) Demonstrate the for loop

Code ->

```
// Demonstrate the for loop.
public class ForTick {

    public static void main(String[] args) {
        int n;

        for (n=10; n>0; n--)
            System.out.println("tick " +n);
    }
}
```

Output ->

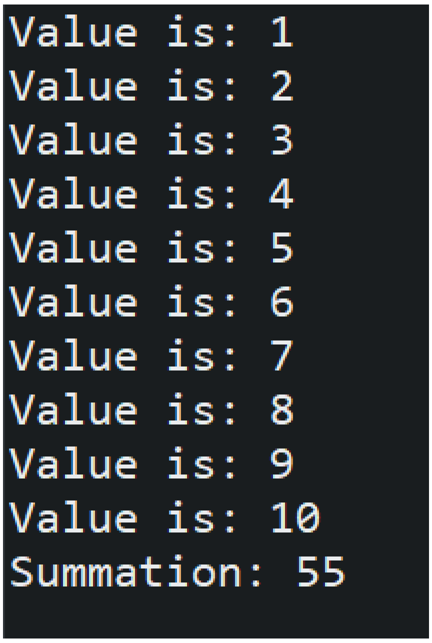


```
tick 10
tick 9
tick 8
tick 7
tick 6
tick 5
tick 4
tick 3
tick 2
tick 1
```

24) The for-each version of the for loop
Code ->

```
// Use a for-each style for loop.
public class ForEach {

    public static void main(String[] args) {
        int nums [] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
        int sum = 0;

        // use for-each style for to display and sum the
values
        for (int x : nums)
        {
            System.out.println("Value is: " + x);
            sum += x;
        }
        System.out.println("Summation: " + sum);
    }
}
```

Output ->

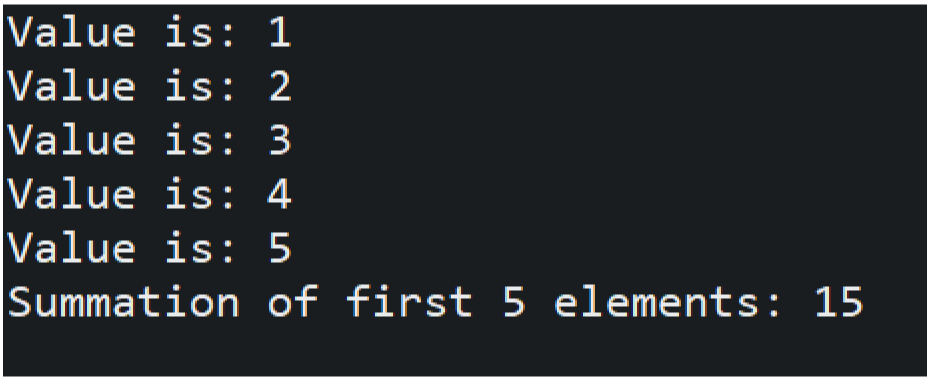


```
Value is: 1
Value is: 2
Value is: 3
Value is: 4
Value is: 5
Value is: 6
Value is: 7
Value is: 8
Value is: 9
Value is: 10
Summation: 55
```

25) Use break with for-each style for
Code ->

```
// Use break with a for-each style for.
public class ForEach2 {

    public static void main(String[] args) {
        int sum = 0;
        int nums [] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

        // use for to display and sum the values
        for (int x : nums)
        {
            System.out.println("Value is: " + x);
            sum += x;
            if (x == 5) break; // stop the loop when 5
is obtained
        }
        System.out.println("Summation of first 5
elements: " + sum);
    }
}
```

Output ->

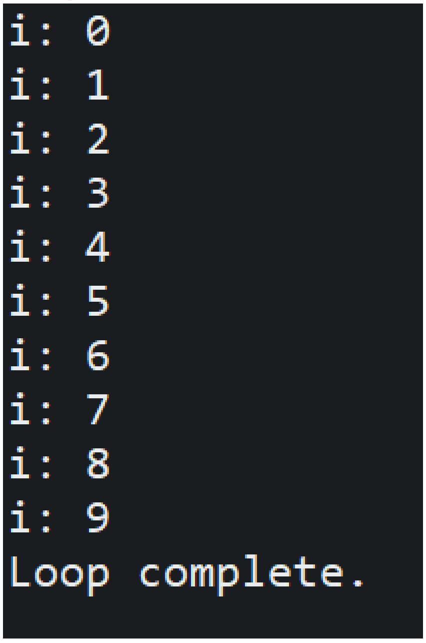


```
Value is: 1
Value is: 2
Value is: 3
Value is: 4
Value is: 5
Summation of first 5 elements: 15
```

26) Jump statements using break to exit a loop
Code ->

```java
// Using break to exit a loop.
public class BreakLoop {

	public static void main(String[] args) {
		for (int i=0; i<100; i++)
		{
			if (i == 10) break; // terminate loop if i is 10
			System.out.println("i: " + i);
		}
		System.out.println("Loop complete.");
	}
}
```

Output ->

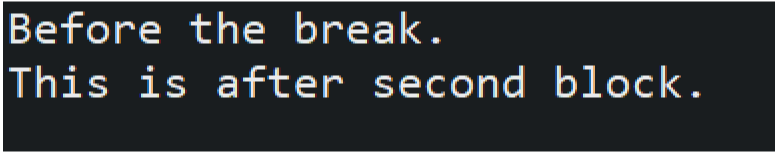


```
i: 0
i: 1
i: 2
i: 3
i: 4
i: 5
i: 6
i: 7
i: 8
i: 9
Loop complete.
```

27) Using break as a form of goto
Code ->

```java
// Using break as a civilized form of goto.
public class Break {

    public static void main(String[] args) {
        boolean t = true;

        first: {
            second: {
                third: {
                    System.out.println("Before the break.");
                    if (t) break second; // break out of second block
                    System.out.println("This won't execute");
                }
                System.out.println("This won't execute");
            }
            System.out.println("This is after second block.");
        }
    }
}
```

Output ->

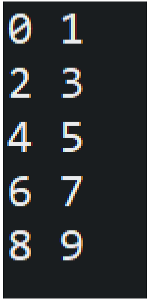


```
Before the break.
This is after second block.
```

28) Using continue
Code ->

```java
// Demonstrate continue.
public class Continue {

    public static void main(String[] args) {
        for (int i=0; i<10; i++)
        {
            System.out.print(i + " ");
            if (i%2 == 0) continue;
            System.out.println("");
        }
    }
}
```

Output ->



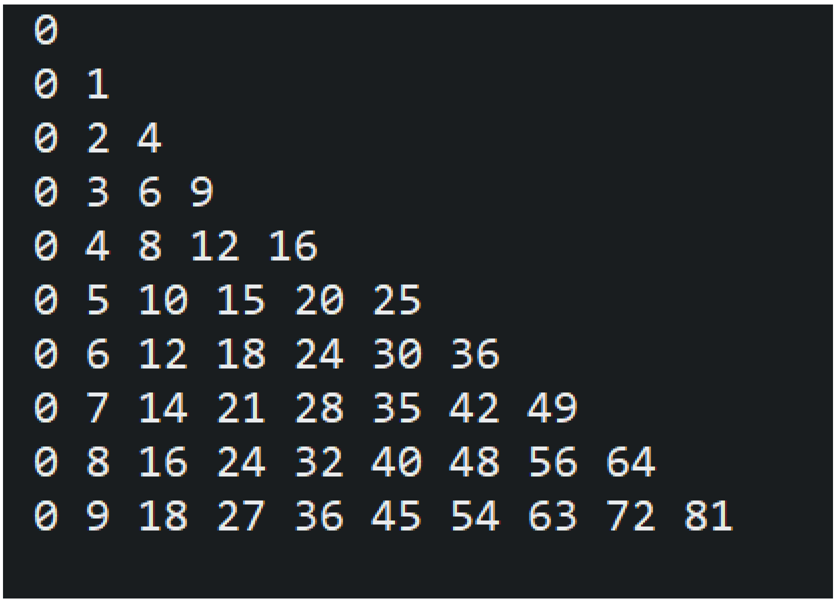
```
0 1
2 3
4 5
6 7
8 9
```

29) Using continue with label

Code ->

```
// Using continue with a label.
public class ContinueLabel {

    public static void main(String[] args) {
        outer: for (int i=0; i<10; i++)
        {
            for (int j=0; j<10; j++)
            {
                if (j > i)
                {
                    System.out.println();
                    continue outer;
                }
                System.out.print(" " + (i * j));
            }
        }
        System.out.println();
    }
}
```

Output ->

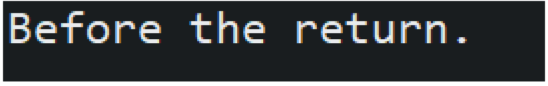


```
0
0 1
0 2 4
0 3 6 9
0 4 8 12 16
0 5 10 15 20 25
0 6 12 18 24 30 36
0 7 14 21 28 35 42 49
0 8 16 24 32 40 48 56 64
0 9 18 27 36 45 54 63 72 81
```

30) Demonstrate Return
Code ->

```
// Demonstrate return.
public class Return {

	public static void main(String[] args) {
		boolean t = true;

		System.out.println("Before the return.");

		if (t) return; // return to caller

		System.out.println("This won't execute.");
	}
}
```

Output ->



```
Before the return.
```