Basics

Ques 1) Print Z

Input Format

There is no input

Output Format

```
*****
   *
  *
 *
*****
```

System.out.println("*****\n___*\n__*\n_*\n*****")

(3 spaces, 2 space, 1 space)

```java
import java.util.*;

public class Main {

  public static void main(String[] args) {
    // Write your code here

    System.out.println("*****\n   *\n  *\n *\n*****");

  }
}
```

Ques) Grading System

1. You are given as input marks of a student.
2. Display an appropriate message based on the following rules:
   - 2.1 for marks above 90, print excellent.
   - 2.2 for marks above 80 and less than equal to 90, print good.
   - 2.3 for marks above 70 and less than equal to 80, print fair.
   - 2.4 for marks above 60 and less than equal to 70, print meets expectations.
   - 2.5 for marks less than equal to 60, print below par.

Note -> Only change the code in area after - // code here

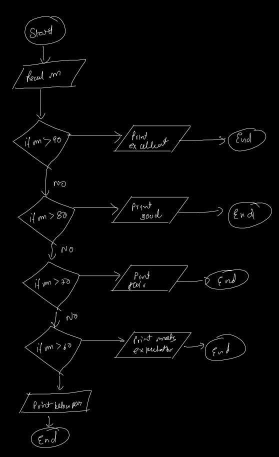
Sample Input

92

Sample Output

excellent

```java
// code here
if(marks > 90) {
    System.out.println("excellent");
} else if(marks > 80) {
    System.out.println("good");
} else if(marks > 70) {
    System.out.println("fair");
} else if(marks > 60) {
    System.out.println("meets expectations");
} else {
    System.out.println("below par");
}
```



Ques 4 All Primes Till N

Sample Input

6
24

Sample Output

7
11
13
17
19
23

i = low ; i <= high ; i++

→ Apply isPrime logic for every i

```java
Scanner scn = new Scanner(System.in);
int low = scn.nextInt();
int high = scn.nextInt();
int i = low;
for(;i<=high;i++) {
    boolean isPrime = true;
    for(int j=2;j*j<=i;j++) {
        if(i%j == 0) {
            isPrime = false;
            break;
        }
    }
    if(isPrime) {
        System.out.println(i);
    }
}
```

Ques) fibonnaci Numbers Till N

n = 10

0	1	1	2	3	5	8	13	21	34
1	2	3	4	5	6	7	8	9	10

$fib(n) = fib(n-1) + fib(n-2)$

i.e n^{th} fibonacci is the sum of previous 2.

1^{st} fib = 0

2^{nd} fib = 1

```java
public static void main(String[] args) {
    // write your code here
    Scanner scn = new Scanner(System.in);

    int n = scn.nextInt();

    int a = 0;
    int b = 1;
    int c = 0;

    for(int i=1;i<=n;i++) {
      System.out.println(a);
      c = a + b;
      a = b;
      b = c;
    }
}
```

Ques) Is a Number Prime

① Brute force Approach

i → 2 to n-1 (if i divides n then no is not prime)
else it is prime

```java
// write ur code here
int t = scn.nextInt();

for(int i=1;i<=t;i = i +1) {

    int n = scn.nextInt();
    boolean isPrime = true;

    for(int j=2;j<n;j++) {
        if(n%j == 0) {
            System.out.println("not prime");
            isPrime = false;
            break;
        }
    }

    if(isPrime) {
        System.out.println("prime");
    }
}
```

→ The codes are for t test case and not just a single no.

② i → 2 to n/2 (if i divides n, no is not prime)
else prime

↑ Because of even nos (think)

```java
// write ur code here
int t = scn.nextInt();

for(int i=1;i<=t;i = i +1) {

    int n = scn.nextInt();
    boolean isPrime = true;

    for(int j=2;j<=n/2;j++) {
        if(n%j == 0) {
            System.out.println("not prime");
            isPrime = false;
            break;
        }
    }

    if(isPrime) {
        System.out.println("prime");
    }
}
```

→ Again, this for t testcases

③ Consider the nos 24 & 36

24 ↑ not a ps; 36 ↑ a perfect sq.

i (24)	i (36)
1 x 24 = 24	1 x 36 = 36
2 x 12 = 24	2 x 18 = 36
3 x 8 = 24	3 x 12 = 36
4 x 6 = 24	4 x 9 = 36
6 x 4 = 24	6 x 6 = 36 → leftout
8 x 3 = 24	9 x 4 = 36
12 x 2 = 24	12 x 3 = 36
24 x 1 = 24	18 x 2 = 36
	36 x 1 = 36

$< \sqrt{n}$ (first four rows for 24) — Mirror images

$<= \sqrt{n}$ (for 36) — Mirror image

↳ To avoid checking mirror images, just check till $<= \sqrt{n}$

So, i → 2 to $\sqrt{n}$ (if i divides n, it is not prime)
else prime.

→ for t testcases

```java
// write ur code here
int t = scn.nextInt();

for(int i=1;i<=t;i = i +1) {

    int n = scn.nextInt();
    boolean isPrime = true;

    for(int j=2;j*j<=n;j++) {
        if(n%j == 0) {
            System.out.println("not prime");
            isPrime = false;
            break;
        }
    }

    if(isPrime) {
        System.out.println("prime");
    }
}
```

Basic Maths & Digit Traversal

Ques 6 Count Digits in a Number

Example
Sample Input
65784383
Sample Output
8

n = 65784383

	n	count	rem
	65784383	0	-
n = n/10	6578438	1	3
n = n/10	657843	2	8
n = n/10	65784	3	3
n = n/10	6578	4	4
n = n/10	657	5	8
n = n/10	65	6	7
n = n/10	6	7	5
n = n/10	0	8	6
n = n/10			

} while (n > 0)

```
int n  = scn.nextInt();

int nod = 0;

while(n > 0) {
    n = n/10;
    nod++;
}

System.out.println(nod);
```

Ques 7 Digits of a Number

Example
Sample Input
65784383
Sample Output
6
5
7
8
4
3
8
3

6 5 7 8 4 3 8 3
↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
7 6 5 4 3 2 1 0 (place values)

initial pv = nod - 1
initial pow = 10^{pv}
while (power > 0)
{ rem = n / 10
n = n % pow;
pow = pow / 10;
}

```
int nod = 0;
int temp =n;
while(temp > 0) {
    temp = temp / 10;
    nod++;
}

int pv = nod-1;
int power = (int)(Math.pow(10,pv));

while(power > 0) {
    System.out.println(n/power);
    n = n%power;
    power /= 10;
}
```

Ques Reverse a No (Display in Reverse)

Example
Sample Input
65784383
Sample Output
3
8
3
4
8
7
5
6

Apply Count Digits logic
just print the digits also.

```
while(n > 0) {
    System.out.println(n%10);
    n = n/10;
}
```

Ques Rotate a no.

Example
Sample Input
562984
2
Sample Output
845629

N → 5 6 2 9 (8 4)
k = 2 84 5629

562984

k = 0	562984
k = 1	456298
k = 2	845629
k = 3	984562
k = 4	298456
k = 5	629845
k = 6	562984

k = -1	629845
k = -2	298456
k = -3	984562
k = -4	845629
k = -5	456298
k = -6	

① k = k % nod
② if (k < 0) k = k + n

562984
↓ k = 2 → 562984 % 10^k = 84
↓ n = n / 10^k
5629

845629
n = 84 × $10^{(nod-k)}$ + n

```
int temp = n;

int nod = 0;
while(temp > 0) {
  temp = temp / 10;
  nod++;
}

k = k%nod;
if(k<0) k += nod;

int power = (int)(Math.pow(10,k));
int part = n % power;
n = n / power;
int sPow = (int)(Math.pow(10,nod-k));
n = part*sPow + n;

System.out.println(n);
```

The visualisation of Rotation

Positive Rotation

↑ 5 ↑ 6 ↑ 2 ↑ 9 ↑ 8 ↑ 4 ↑
6 5 4 3 (2) 1 0

↳ The nos before this pos will come at the front with the same seq

k = 2

Rotated Number:
↑ 8 ↑ 4 ↑ 5 ↑ 6 ↑ 2 ↑ 9 ↑
6 5 4 3 2 1 0

Negative Rotation

↑ 5 ↑ 6 ↑ 2 ↑ 9 ↑ 8 ↑ 4 ↑
0 -1 (-2) -3 -4 -5 -6

Nos before this pos will go at the back with same seq.

Rotated No:
↑ 2 ↑ 9 ↑ 8 ↑ 4 ↑ 5 ↑ 6 ↑
0 -1 -2 -3 -4 -5 -6

Ques Inverse of a No

Example
Sample Input
28346751
Sample Output
73425681

2 8 3 4 6 7 5 1 (face values)
↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
8 7 6 5 4 3 2 1 (place values)

$$n = \sum_{i=1}^{nod} fv * 10^{pv-1}$$

int pv = 1;
fv = n % 10
n = n / 10 → swap pv & fv
ans = pv * 10^{fv-1}

```
int pv = 1;
int ans = 0;
while(n > 0) {

    int fv = n%10;
    n = n/10;

    ans += pv * Math.pow(10,fv-1);
    pv++;
}

System.out.println(ans);
```

Ques) GCD and LCM

GCD is the HCF actually.

lcm = (n1 * n2) / GCD

Sample Input
36
24
Sample Output
12 → GCD
72 → LCM

GCD: Greatest Common Divisor
i.e the greatest no that can perfectly divide both the given nos.

```
int n1 = scn.nextInt();
int n2 = scn.nextInt();

int on1  = n1;
int on2  = n2;
while(n1 % n2 > 0) {
    int rem = n1 % n2;
    n1 = n2;
    n2 = rem;
}

System.out.println(n2);

System.out.println((on1*on2)/n2);
```

24) 36 (1
−24
12) 24 (2
−24
0
(12 → gcd)

36) 24 (0
−0
24) 36 (1
−24
12) 24 (2
−24
0
(12 → gcd)

while (n1 % n2 > 0)
{ rem = n1 % n2;
n1 = n2;
n2 = rem;
}
ans = n2;

Ques) Prime factorisation

Example
Sample Input
1440
Sample Output
2 2 2 2 2 3 3 5

2	1440
2	720
2	360
2	180
2	90
3	45
3	15
5	5
	1

fac = 2;
↳ keep dividing till no is divisible

```
int factor = 2;
while(n > 1) {

    if(n%factor == 0) {
        System.out.print(factor + " ");
        n = n/factor;
    } else {
        factor++;
    }
}
```

Ques) Curious Case of Benjamin Bulbs

24

1 × 24
2 × 12
3 × 8
4 × 6
} On in 1st fluctuation

6 × 4
8 × 3
12 × 2
24 × 1
} Off in 2nd fluctuation

Since all the non perfect square have even factors, they will always be off

36

1 × 36
2 × 18
3 × 12
4 × 9
6 × 6 → one extra factor causes perfect square bulbs to be on.
9 × 4
12 × 3
18 × 2
36 × 1

```
int n = scn.nextInt();

for(int i=1;i*i<=n;i++) {

    System.out.println(i*i);
}
```

Ques) Pythagorean Triplets

$a^2 + b^2 = c^2$

Eg: $3^2 + 4^2 = 5^2$

```
int a = scn.nextInt();
int b = scn.nextInt();
int c = scn.nextInt();

if(a*a + b*b == c*c) {
    System.out.println(true);
} else if(b*b + c*c ==a*a) {
    System.out.println(true);
} else if(a*a + c*c == b*b) {
    System.out.println(true);
} else {
    System.out.println(false);
}
```