Number Patterns

1)

```
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
```

Ans

row (i)	column (j)
1	1 2 3 4 5
2	1 2 3 4 5
3	1 2 3 4 5
4	1 2 3 4 5
5	1 2 3 4 5

Prog

```
class Demo
{
    PSVM(String args[])
    {
        Scanner S = new Scanner(System.in);
        int n = S.nextInt();

        for(i=1; i<=n; i++)
        {
            for(j=1; j<=n; j++)
            {
                S.O.P(j);
            }
            S.O.Pln();
        }
```

2> 1 1 1 1 1
2 2 2 2 2
3 3 3 3 3
4 4 4 4 4
5 5 5 5 5

→

row(i)	column(j)
1	1 1 1 1 1
2	2 2 2 2 2
3	3 3 3 3 3
4	4 4 4 4 4
5	5 5 5 5 5

Prog

```
for(int i=1 ; i<=n ; i++)
  {
   for (int j=1 ; j<=n ; j++)
     {
       S.O.P(j);
     }
    S.O.Pln();
  }
```

3>

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

row(i)	colm(j)
1	1
2	1, 2
3	1 2 3
4	1 2 3 4

```
for(i=1 ; i<=n; i++)
 {
   for(j=1 ; j<=i ; j++)
  {
     Sop("*");
  }
```

4)

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

row(i)	col(j)
1	1
2	2
3	3
4	4

Prog

```
for(i=1 ; i<=n ; i++)
{
  for(j=1 ; j<=i ; j++)
  {
    S.O.P(j);
  }
  sopln();
}
```

5)

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

row(i)	col(j)
1	1
2	2
3	3
4	4

Prog

```
for(i=1 ; i<=n ; i++)
{
  for(j=1 ; j<=i ; j++)
  {
    S.O.P(j);
  }
  S.Opln();
}
```

*	5 4 3 2 1	for(i)	col(j)
	5 4 3 2 1	1	5 4 3 2 1
	5 4 3 2 1	2	5 4 3 2 1
	5 4 3 2 1	3	5 4 3 2 1
	5 4 3 2 1	4	$\eta >= 1$
		5	

Prog

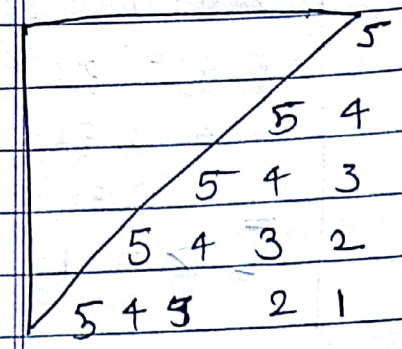
```
for( i=1 ; i<=n ; i++)
 {
  for(j=n ; j>=1 ; j--)
   {
     S.O.P(j);
   }
   S.O.Plm();
 }
}
```

*	5	for(i)	col(j)
	5 4	1	5
	5 4 3	2	5 4
	5 4 3 2	3	5 4 3
	5 4 3 2 1	4	5 4 3 2
		5	5 4 3 2 1

Prog

```
for( i=1 ; i<=n ; i++)
 {
 int count=n;
  for(j=1 ; j<=i ; j++)
  {
    S.O.P(count);
    count--;
  }
  S.O.plm();
}}
```

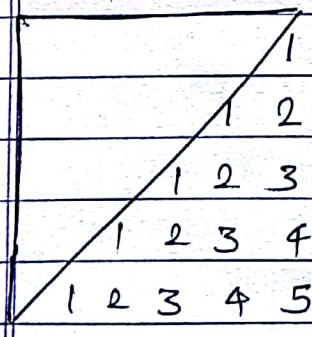
*



Prog

```
for(i=1 ; i<=n ; i++)
{
for(j=1 ; j<=n-i ; j++)
  {
    S.O.P ("_");
  }
int count = n;
for(j=1 ; j<=i ; j++)
  {
    sop (count);
    count--;
  }
sopln();
}
}
```

*




```
for(i=1 ; i<=n ; i++)
{
for(j=1 ; j<=n-i ; j++)
  {
    sop ("_");
  }
for(j=1 ; j<=i ; j++)
  {
    sop(j);
  }
} sopln();
}
```

#

1	1	1	1
0	0	0	0
1	1	1	1
0	0	0	0

row(i)	col(j)
0	1 1 1 1
1	0 0 0 0
2	1 1 1 1
3	0 0 0 0
4	

Prog

```
for(i=1 ; i<=n ; i++)
{
  for(j=1 ; j<=n ; j++)
  {
    if(i%2==0)
    {
      sop("0");
    }
    else
    {
      sop("1");
    }
    soplm();
  }
}
```

#

1	0	1	0	1
1	0	1	0	1
1	0	1	0	1
1	0	1	0	1

```
for(i=1 ; i<=n ; i++)
{
  for(j=1 ; j<=n ; j++)
  {
    if(j%2==0)
    {
      S.O.P("0");
    }
    else
    {
      S.O.P("1");
    }
  }
}
```

#

```
1 0 1 0 1 0
0 1 0 1 0 1
1 0 1 0 1 0
0 1 0 1 0 1
```

```
for(i=1; i<=n; i++)
{
  for(j=1; j<=n; j++)
  {
    if((i+j)%2!=0)
    {
      sop("0");
    }
    else
    {
      sop("1");
    }
    soplm();
  }
}
```

1+1=2%2==0

1 0 1

(1,1) (1,2) (1,3)

0 1 0

#

```
1
0 1
1 0 1
0 1 0 1
```

```
for(i=1; i<=n; i++)
{
  for(j=1; j<=i; j++)
  {
    if((i+j)%2==0)
    {
      sop("1");
    }
    else
    { sop("0");
    }
```

```
        SOPln();
    }
}
```

#

```
1 1 1 1 2
3 2 2 2 2
3 3 3 3 4
5 4 4 4 4
5 5 5 5 6
```

proof

```
for (int i=1 ; i<=n ; i++)
{
    if ( i%2==0)
    {
      S.OP( i+1);
    }
for (j=1 ; j<n ; j++)
  {
    SOP(i);
  }
  if (i%2 !=0)
    {
      S.OP (i+1);
    }
  SOPln();
}
}
```

#

1

3 2

6 5 4

10 9 8 7

⇒

```
int k=1;
for(i=0 ; i<n ; i++)
{
  for(j=0 ; j<n-i+1 ; j++)
  {
    SOP("_");
  }
 for( j=0 ; j<=i ; j++)
  {
   SOP(k-j);
  }
   SOPln();
   k=k+i+2;
  }
}
```

#

1

3*2

6*5*4

10*9*8*7

⇒

```
int k=1;
for(int i=0 ; i<n ; i++)
  {
  for(j=0 ; j<2*(n-i-1) ; j++)
    {
      SOP("_");
    }
```

```
for(j=0; j<=i; j++)
{
  if(j<i)
  {
    S.OP((k-j)+"*");
  }
  else
  {
    SOP((k-j));
  }
}
SOPln();
k=k+i+2;
}
```

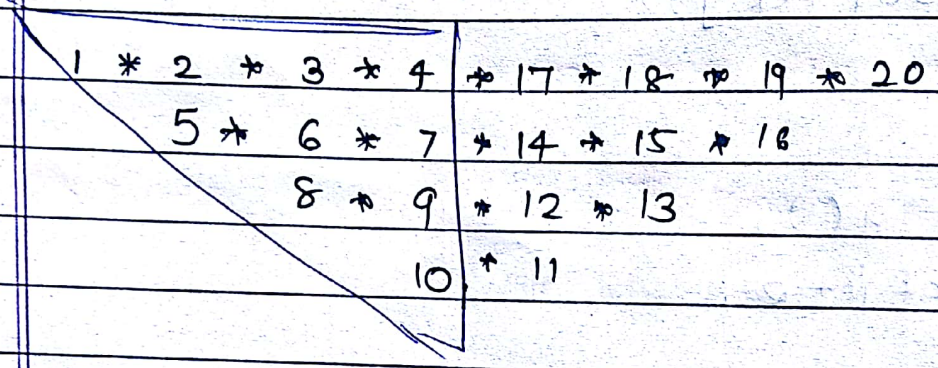
#

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

```
for(int i=1; i<=n; i++)
{
  if(i%2!=0)
  {
    count+=i;
  }
  else
  {
    count+=i-1;
  }
for(j=1; j<=i; j++)
{
  if(i%2==0)
  {
    SOP(count);
    count--;
  }
```

```
        else
        {
        sop(count);
            count++;
        }
    }
    s.o.println();
    }
}
```

#




```
int b = n*n+1;
int count = 1;
int k = 0;
int p = n-1;

for(int i=0; i<n; i++)
  {
  for(int j=1; j<=i; j++)
    {
    sop("  ");
    }
  for(j=0; j<n-i; j++)
    {
    sop(count+"* ");
    count++;
    }
```

```
for (int j=0 ; j<n-i ; j++)
 {
    if (j==0)
     {
       k=b ;
     }
else if (j<n-i-1)
    {
      S.O.P (b+"*");
       b++ ;
    }
else
   {
      sop(b);
   }
 }
  b = k-p ;
    p--;
    sopln() ;
  }
 }
}
```

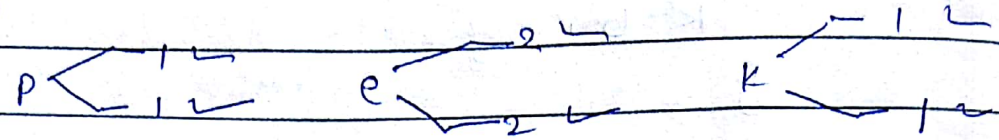
★

```
--1--2--3--4--5
16--17--18--19- 6
15   24 - 25  20  7
14   23 - 22 -- 21  8
13 - 12--11--10--9
                 7
```

WAP to check the string is angram or not

String 1 : peek

String 2 : keep



ALGORITHM

1) Remove space from both the strings

2) Check lengths —Not same→ "NOT ANAGRAM"
Same ↓

3) Convert both the string either Uppercase or lower case

4) sort both the string

5) compare

```
class Anagram
{
    string removespace(String s)
    {
        String temp = " ";
        for(int i=0 ; i<s.length() ; i++)
        {
            if(s.charAt(i)
```

WAP to check whether the given input string is PANAGRAM or not.

Algorithm

1) Concern only [A-Z, a-z] → (65-90 || 97-122)

2) Convert either Uppercase or lowercase

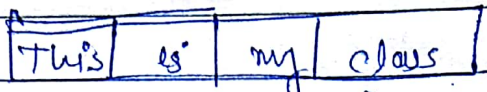
3) sort the string

4) Remove duplicates

5) Check length → if length >= 26 → Panagram
↓
Not panagram

☆ String to String type Array

"This is my class"




```
int count = 0;
for (int i=0 i< s.length() ; i++)
 {
   if ( s.charAt (i) == ' ' && s.charAt (i+1) != " ")
    {
      count ++;
    }
String [] ar = new String [count+1];
int j=0;
String temp = " ";

for (int i=0 ; i<s.length(); i++)
    {
      if (s.charAt(i) != ' ')
          {
            temp += s.charAt(i);
          }
else if (s.charAt(i) == ' ')
  {
    ar[j] = temp;
     j++;
     temp = " ";
    }
}
ar[j] = temp;
```

last : 21

```
for (int i=0 ; i< ar.length ; i++)
  {
    Sop a[i]);
  }
```

Case 1 except : class any is this

```
temp = " ";
for (int i = ar.length-1 ; i>0 ; i--)
  {
    temp += ar[i] + " ";
  }
temp += ar[0];
```

Case 2 Shit si ym ssalc

```
temp = " ";
for (int i=0 ; i<ar.length ; i++)
  {
    ar[i] = reverse (ar[i])
  }
for (int i=0 ; i<ar.length-1 ; i++)
  {
    temp = ar[i]+" ";
  }
  temp += ar[ar.length-1];
    sop(temp);

Static String reverse (String s)
{
  String t = " ";
  for (int i = s.length()-1 ; i>=0 ; i--)
    {
      t += s.charAt(i);
    }
  return t;
}
```

Case 3 This is2 my2 class

```
temp = " ";
for ( int i = 0 ; i < ar.length - 1 ; i++ )
{
    temp += ar[i] + (ar[i]).length() + " ";
}
temp += ar[ar.length - 1] + (ar[ar.length - i]).
                                length();
Sop(temp);
```

4 This Is My class

```
temp = " "
for ( int i = 0 ; i < ar.length ; i++)
{
    ar[i] = conversion (ar[i]);
}
for (int i = 0 ; i < ar.length - 1 ; i++ );
{
    temp += ar[i] + " ";
}

Static string conversion ( String s )
{
    String s = " ";
if ( s.charAt(0) >= 97 && s.charAt(0) <= 122)
{
    t += char (s.charAt(0) - 32);
}
```

```
else
{
    t += s.charAt(0);
}
for (int i = 1 ; i < s.length() ; i++)
{
    t += s.charAt(i);
}
return t;
```

case 5 This is my class
Class This 9 my is 4

```
temp = " ";

for (int i=0 ; i< ar.length/2 ; i++)
{
  temp += ar[ ar.length-1 -i] + ar[i] +
        ( ar [ar.length-1-i] + ar[i]).length()+" ";
}

if (ar.length %2 !=0 )
{
  temp += ar[ar.length/2 ] + (ar[ar.length/2]).
                              length();
}
sop(temp);
```

A	B	C	D	E
65	66	67	68	69
F	G	H	I	J
70	71	72	73	74
K	L	M	N	O
75	76	77	78	79
P	Q	R	S	T
80	81	82	83	84
U	V	W	X	Y	Z
85	86	87	88	89	90

→ ABC key = 4

RFG

XYZ

BCD

(1) 65 − (90 − key) + 1

(2) key

X → 4 → B

88 66

65 + key − ((90 − s.charAt(i)) − 1);

// String temp = " ";

key = 4;

```
for (int i=0 ; i<s.length() ; i++)
{
    if (s.charAt(i) >= 65 && s.charAt(i) <= (90-key))
    {
        temp += (char)(s.charAt(i) + key);
    else
```

```
{
    temp += (char)(65+key-(90-s.charAt(i))-1);
  }
}
else if (s.charAt(i)>=97 && s.charAt(i)<=122)
{
  if (s.charAt(i)>=97 && s.charAt(i)<=(122-key))
    {
      temp += (char)(s.charAt(i)+key);
  else
    {
      temp += (char)(97+key-(122-s.charAt(i))-1);
    }
}
else if (s.charAt(i)>=48 && s.charAt(i)<=57)
{
  if (s.charAt(i)>=48 && s.charAt(i)<=57-key)
    {
      temp += (char)(s.charAt(i)+key);
    }
  else
    {
      temp += (char)(48+key-(57-s.charAt(i)-1);
    }
}
else
  {
    temp += s.charAt(i);
  }
}
  sop(temp);
}
```

Assignment

⇒ WAP to find out $A \cup B$, $A \cap B$, $A - B$, $B - A$

⇒ WAP to find Occurring & Frequency

1	2	3
00	01	02
4	5	6
10	11	12
7	8	9
20	21	22

90° right →

7	4	1
00	10	00
8	5	2
21	11	01
9	6	3
22	12	02

```
for (int i=0 ; i<ar.length ; i++)
{
  int k=ar.length-1;
for(int j=0 ; j<ar[i].length ; j++)
  {
    ar1[i][j] = ar[k][i] ;
      k--;
  }
}
```

1	2	3
4	5	6
7	8	9

90° left →

3	6	9
2	5	8
1	4	7

```
int k=ar.length-1;
for(int i=0 ; i<ar.length ; i++)
{
for(int j=0 ; j<ar[i].length ; j++)
  {
    a[i][j] = ar[j][k];
  }
  k--;
}
```

*	ArrayList	Vector
⇒	There are not synchronized	⇒ These are synchronized
⇒	These are increments 50 % of current array size, if no of element exceeds from it's capacity	⇒ Vector increments 100% means double the array size if total no of element exceed than it's capacity
⇒	Not a legacy class	⇒ It's a legacy class
⇒	Fast	⇒ slow

	ArrayList	LinkedList
is	Internally uses dynamic array to	
	HashMap	HashTable
*	It is non-synchronized it is not thread safe and can't be shared between many threads without proper synchro. code	⇒ It is synchronized it is thread safe and can be shared with many threads
#	It's allows null key and multiple null values	⇒ Hashtable doesn't allow nullkey and value

New class introduced in JDK 1.2 Fast	# it is a legacy class # slow
Can be synchronized by calling map on collections. synchronized map(hashmap)	# Internally synchronized and can't be used for synchronized
It is traversed by iterator	# traversed by enumerator & iterator
Iterator in HashMap is fail fast	# Enumerator in this is not fail fast

Collection	Collections
> Collection is an interface for most collection classes.	ii > Collections is a class which has some static methods that method return the collection