```
int isPrimeProduct(int n)
{
    for(int i=2; i<=n/2; i++)
    {
        for(int j=2; j<=n/2; j++)
        {
            if(isPrime(i)==1 && isPrime(j)==1)
            {
                if(i*j==n)
                {
                    return 1;
                }
            }
        }
        return 0;
    }
```

22 is primeproduct = 2x11 > 1

```
int isPrime(int n)
{
    for(int i=2; i<n; i++)
    {
        if(n%i==0)
        return 0;
    }
    return 1;
}
```

2

(odd == even)

a[0]	a[1]	a[2]	a[3]
2	9	5	7
e	o	e	o

```
int isBalanced(int[] a)
{
    for(int i=0; i<a.length; i++)
    {
        if(i%2==0 && a[i]==1)
        return 0;
        if(i%2==1 && a[i]==0)
        return 0;
    }
    return 1;
}
```

3

```
int isCentered(int[] a)
{

int center = a.length/2;
if (a.length%2 == 0)
   return 0;

for(int i=0; i<a.length; i++)
  {

     if (center >= a[i])
      return 0;

   }

  return 1;
}
```

4

```
boolean hasSmallFactors(int n, int k)
{
  for(int i=2; i<k; i++)
   {
     for(int j=i+1; j<k; j++)
       {
          if(i*j == n)
            return 1;

       }
     }
     return 0;

}
```

⑤

fill({9,2,8,5,9,11,-2,19,3,16}, 4, 11)

new array

```
int[] fill(int[] arr, int k, int n)
{
  int[] a = new int[n];
  for(int i=0; i<n; i++)
    {
      a[i] = arr[index];
       index++;
      if(index >= k)
        index = 0;
    }
   return a;
}
```

⑥

{1,2,4,0,0,3,4,3} is Hollow

3 or more 0 in middle, followed by some number of non-zero elements

```
int isHollow(int[] a)
{
 for(int i=0; i<a.length; i++)
    {
      if(a[i] == 0)
        startCount++;
      else
        break;
    }
 for(int i=a.length; i>=0; i--)
    {
      if(a[i] == 0)
        endCount++;
      else
        break;
    }
 for(int i=0; i<a.length; i++)
    {
      if(a[i] == 8)
        count8++;
    }
 if(startCount + endCount + count8 != a.length)
   return 0;
 if(startCount != endCount)
   return 0;
 if(count8 < 3)
   return 0;
 else
   return 1;
}
```

(1) 13013 = 1 * 7 * 11 * 13 (6 3 2) min = 2

```
int minDistance (int n)
{
  int prefactor = n*(-1);
  int min = n;
  for(int i=2; i<n/2; i++)
      {
        if(n%i==0)
          {
            diff = i - prefactor;
            if(min > diff)
                min = diff;
              prefactor = i;
          }
        }return min;
      }
```

* 13013 = 1 * 7 * 11 * 13, max = 12

```
int maxDistance (int n)
{
int maxfactor;
int minfactor;
for(int i=2; i<n; i++)
{
   if(n%i==0)
     {
       minfactor = i;
       break;
     }
}
for(int i=n-1; i>=2; i--)
   {
     if(n%i==0)
     {
       maxfactor = i;
      break;
     }
}
diff = maxfactor - minfactor;
if(maxfactor < diff)
    maxfactor = diff;

return maxfactor;
}
```

8

```
int isWave (int[] a)
{
  for(int i=0; i<a.length; i++)
  {
    if (a[i]%2==0 && a[i+1]%2==0)
      return 0;
    if(a[i]%2==1 && a[i+1]%2==1)
      return 0;
  }
  return 1;
}
```

{7, 2, 9, 10, 5} = wave arr
0 1 0 1 0

9

```
int isBean (int[] a)
{
  for(int i=0; i<a.length; i++)
  {
    if(a[i]==9)
      isfound9=1;
    if(a[i]==18)
      isfound18=1;
    if(a[i]==7)
      isfound7=1;
    if(a[i]==11)
      isfound11=1;
  }
  if(isfound9==0 && isfound18==1 || isfound9==1 && isfound18==0)
    return 0;
  if(isfound7==1 && isfound11==1)
    return 0;
  else return 1;
}
```

9 % 3 = 1
7 % 11 = 1
{9, 6, 18} = 0
a[i] % 6 = 0;

(62)

n	digit	return
32121	1	count
n<0 || digit<0		-1

(63) maxOccurDigit(33321) returns

```
int maxOccurDigit(int n)
{
    while (n!=0)
    {
        if(n%10==digit)
            count++;
        n=n/10;
    }
    return count;
}
```

```
int countDigit(int n, int digit)
{
    if(n<0 || digit<0)
        return -1;
    while(n!=0)
    {
        if(n%10==digit)
            count++;
        n=n/10;
    }
    return count;
}
```

(64)

at least one odd no. comes follow by prime

{4, 9, 6, 7, 3}

```
int isBunkerdong(int[] a)
{
    for(int i=0; i<a.length; i++)
    {
        if(a[i]%2==1 && isPrime(a[i+1])==1)
            return 1;
    }
    return 0;
}
```

(12)

```
int isMeera(int[] a)
{
  for(int i=0; i<a.length; i++)
  {
    for(int j=0; j<a.length; j++)
    {
      if(a[i] == 2 * a[j])
        return 1;
    }
  }
  return 0;
}
```

n = n*2

{1, 4, 2, 8} → 0

n = 4, 2*n = 8

n = 2n

4 = 2 * 4

= 8 ✓

19 Nontrivial factors

```
int isMeera(int n)
{
  for(int i=0; i<n/2; i++)
  {
    for(int i=2; i<=n/2; i++)
    {
      if(n%i == 0)
        count++;
    }
    if(n%count == 0)
      return 1;
    else
      return 0;
  }
```

30 = 2, 3, 5, 6, 10, 15 = 6 nontrivial factors

21 = 3 * 7 = nontrivial factors

30/2 = 15

21/2 = 10

14. {3, 7, 1, 8, 1} ⇒ Bunker array

```
int isBunker(int[] a)
{
    for(int i=0; i<a.length; i++)
    {
        if(isPrime[a[i]] == 1)
            isFoundPrime = 1;
        if(a[i] == 1)
            isFound1 = 1;
    }
    if(isFoundPrime == 1 && isFound1 == 0 ||
       isFoundPrime == 0 && isFound1 == 1)
        return 0;
    else
        return 1;
}
```

15. {2, 10, 9, 3} = 1
{3, 4, 5, 7} = 1
} n+1, n-1

```
int isNice(int[] a)
{
    for(int i=0; i<a.length; i++)
    {
        for(int j=0; j<a.length; j++)
        {
            if(a[i] == a[j]+1 || a[i] == a[j]-1)
                isFound = 1;
        }
        if(isFound == 0)
            return 0;
    }
    return 1;
}
```

Continuous factor = 120, 99
! " = 99, 121

16

```
int isContinuousFactored(int n)
{
  int mul = 1;
  for(int i=2; i<n; i++)
    {
      for(int j=i; j<n; j++)
        {
          mul = mul * j;
          if(mul == n)
            return 1;
          if(mul > n)
            break;
        }
    }
  return 0;
}
```

Prime Twoin no.
8 - 11 = 2
5 - 7 = 2
7 - 5 = 2

17

```
int isTwoin(int[] arr)
{
  for(int i=0; i<arr.length; i++)
    {
      if(isPrime(arr[i]) == 1)
        {
          for(int j=0; j<arr.length; j++)
            {
              if(isPrime(arr[j]) == 1)
                {
                  if(a[i]-a[j] == 2 || a[j]-a[i] == 2)
                    return 1;
                }
            }
        }
    }
  return 0;
}
```

18

```
int isSetEqual(int[]a, int[]b)
{
  for(int i=0; i<a.length; i++)
  {
    for(int j=0; i<b.length; j++)
    {
      if(a[i] == b[j])
        isfound = 1;
    }
    if(isfound == 0)
      return 0;
  }
  return 1;
}
```

if every element in one is also in the other and vice-versa.

{1,9,12} {1,12,9} {12,9,1} {1,9,12,1}
a b a b

19 Guthrie / Smart / Bunker

```
int isSmart(int n)
{
  int seq = 1;
  int gap = 1;
  while(n != 0)
  {
    if(seq == n)
      return 1;
    if(seq > n)
      return 0;
      seq = seq + gap;
      gap++;
  }
}
```

seq = 1, 2, 4, 7, 11, 16 ---
gap = 1 2 3 4 5 . ---

20

```
int isNiceArray(int[] a)
{
  for(int i=0; i<a.length; i++)
  {
    if(isPrime(a[i])==1)
    {
      sum = sum + a[i+1];
    }
  }
    if(sum != a[0])
      return 0;
    else
      return 1;
  }
```

Sum of prime no = first element of array a[0]

{21, 3, 7, 9, 11, 4, 15} = 3, 7, 11 = 21

prime = 0 2 3

21

```
int isComplete(int[] a)
{
  for(int i=0; i<a.length; i++)
  {
    if(a[i]<0)
      return 0;
    if(a[i]%2==0)
    { maxEven=i;
    }
    if(maxEven<a[i])
    { maxEven = a[i];
    }
    for(int i=2; i<maxEven; i=i+2)
    {
      for(int j=0; j<a.length; j++)
      {
        if(i==a[j])
          isFound = 1;
      }
      if(isFound==0)
        return 0;
  } return 1;
```

② a[i] > 0

① all even no < maxEven

a = a[i] > 0

b = maxEven

c = i = a[j]

22

If They have Same no of factors = 1

10 & 130 = 4 factors

```
int factorEqual(int n, int m)
{
   for(int i=0; i<n; i++)
     {
        if(n%i==0)
           count1++;
     }

   for(int i=0; i<m; i++)
     {  if(m%i==0)
           count2++;
     }
   if(count1 != count2)
     return 0;
   else
     return 1;
}
```

23

(a) a[i] < i

(b) sum = 0

```
int isMeera(int[] a)
{
   for(int i=0; i<a.length; i++)
     {
        if(a[i]>i)
          return 0;

        sum = sum + a[i];

        if(sum != 0)
           return 0;
     }

   return 1;
}
```

24 Triple Array / Dual Array / good Spread

every values occurs exactly 3 times

{3, 1, 2, 1, 3, 1, 3, 2, 2}

```
int isTriple(int[] a)
{
  for(int i=0; i<a.length; i++)
  {
    for(int j=0; j<a.length; j++)
    {
      if(a[i] == a[j])
        count++;
    }
    if(count != 3)
      return 0;
  }
  return 1;
}
```

25

Seq = 1, 1, 2, 3, 5, 8, 13, 21

```
int isFibonacci(int n)
{
int a=1, b=1; c;
~~while(n!=0)~~ for(int i=0; i<n-2; i++)
  {
    c = a+b;
    a = b;
    b = c;
    if(c==n)
      return 1;
    ~~else~~ if(c>n)
    {     return 0;
}
```

26 Fancy number

Seq = 1, 1, 5, 17, 61 ...
(a, b)

3×a + 2×b = c
3×1 + 2×1 = 5

```
int isFancy(int n)
{
    int a=1, b=1, c;
    while(n!=0) for(int i=0; i<n-2; i++)
    {
        c = 3*a + 2*b;
        a = b;
        b = c;
        if(c==n)
            return 1;
        if(c>n)
            return 0;
    }
}
```

27.

```
int isMeera(int[] a)
{
  for(int i=0; i<a.length; i++)
  {
    if(a[i]==1)
        isfound1 = 1;
    if(a[i]==9)
        isfound9 = 1;
  }
  if(isfound1==0 && isfound9==1 || isfound1==1 && isfound9==0)
    return 0;
  else
    return 1;
}
```

(19) = {7, 9, 0, 10, 1} - 1
{6, 10, 8} - 1
{7, 6, 1} - 0
{9, 10, 0} - 0

2n, 2n+1 or n/2

28

```
int isBean(int[] a)
{
  for(int i=0; i<a.length; i++)
  {
    for(int j=0; j<a.length; j++)
    {
      if(a[i]==2*a[j] || a[i]==2*a[j]+1 || a[i]==a[j]/2)
        isfound = 1;
    }
    if(isfound==0)
      return 0;
  }
  return 1;
}
```

4, 9, 8 : 2×4 = 8
2×4+1 = 9
4/2 = 2

29

```
int isoddHeavy(int[] a)
{
  for(int i=0; i<a.length; i++)
  {
    if(a[i]%2==1)
    {
      if(oddmin > a[i])
        oddmin = a[i];
      else
        if(evenMax < a[i])
          evenMax = a[i];
    }
    if(oddmin < evenMax)
      return 0;
    else
      return 1;
  }
```

oddmin evenmax

{11, 4, 9, 2, 8} - 1

{11, 4, 9, 2, 3, 10} - 0

31 int isNormal(int n)

```
{
    for(int i=0; i<n; i++)
    {
        if(n%i==0)
        {
            if(i%2==1)
                return 0;
        }
    }
    return 1;
}
```

no odd factors except for 1

1, 2, 3, 4, 5, 7, 8 → 1

6, 9, 10 → 0

2×3, 3×3, 2×5 → odd factors

32

```
int isAllPossibilities(int[] a)
{
    for(int i=0; i<a.length; i++)
    {
        for(int j=0; j<a.length; j++)
        {
            if(a[i]==i)
                return 1; // isfound = 1;
        }
    }
    return 0;
}
```

→

```
if(isfound==0)
{ return 0;
  return 1;
}
```

a[i] == i

{0, 1, 2, 3, 4}

a[0] = 0

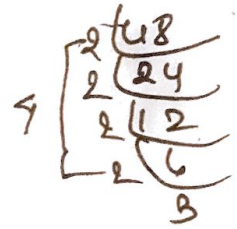
a[1] = 1

a[2] = 2

(31)
```
int isDual(int[] a)
{
int sum = a[0]+a[1];
for(int i=0; i<a.length; i++)
  {
    if(a[i]+a[i+1] != sum)
      return 0;
  }
return 1;
}
```

{1,2, 3,0} = 1
{1,2, 2,1, 3,0} = 1
{1,1, 2,2} = 0

(33)
```
int factorTwoCount(int n)
{
    while(n!=0)
    {
        if(n%2==0)
            count++;
        n = n/2;
    }
    return count;
}
```



return 4

(34)

```
int isDaphne(int[] a)
{
  for(int i=0; i<a.length; i++)
    {
      if(a[i]%2==1)
          isfoundodd=1;
    }
  for(int i=0; i<a.length; i++)
    {
      if(a[i]%2==0)
          countBeginsEven++;
    }
  for(int i=a.length-1; i>=0; i--)
    {
      if(a[i]%2==0)
          countEndsEven++;
    }
  if(countBeginsEven!=countEndsEven && isfoundodd==1 ||
     countBeginsEven==countEndsEven && isfound==0)
      return 0;
  else
      return 1;
}
```

{4, 8, 6, 3, 2, 9, 8, 1}, {8, 13, 12, 12, 6}

① begins & ends same no. of even
② at least one odd

(35)

```
int sumDigits(int n)
{
 int sum=0;
   while(n!=0)
    {
      int dig = n%10;
          n = n/10;
          sum = sum+dig;
    }
   return sum;
}
```

8114 = 9(8+1+4+4)

36

```
int isPascal(int n)
{
    for(int i=0; i<n; i++)
    {
        sum = sum + i;
    }
    if(sum==n)
        return 1;
    else
        return 0;
}
```

Sum of integers from 1 to j

1+2 = 3

1+2+3 = 6

1+2+3+4 = 10

1+2+3+5 = 15

if(i*(i+1)/2 == n)

(37)

```
boolean sumIsPower(int[] arr)
{
    for(int i=0; i<arr.length; i++)
    {
        sum = sum + arr[i];
    }
    for(int i=0; i<sum; i++)
    {
        pow = pow*2;
        if(sum==pow)
            return 1;
        if(sum>pow)
            return 0;
    }
}
```

Sum = 2^n (1, 2, 4, 8, 16)

1, 2, 4, 8, 16 = 8+8+8+8 = 32 = 2^5

if(pow==sum)
return true;
if(pow>sum)
break;
}
return false;
}

30

digit are all even 2421 = 1

```
int isRiley (int n)
{
    while(n != 0)
    { int dig = n%10;
      n = n/10;
      if ( dig%2 == 1)
        return 0;
    }
    return 1;
}
```

31

index of the lasteven value

{3, 2, 5, 6, 7}

i = 3 value = 6

```
int lastEven (int[] a)
{
  for(int i = a.length-1; i >= 0; i--)
    {
      if (a[i]%2 == 0)
       return i;
    }
}
```

countmax = {6, 9, 1, 9, 4, 5, 1, 5} ∈ 9 = 2

```
int countMax(int[] a)
{
  for(int i=0; i<a.length; i++)
  {
    if(max < a[i])
      max = a[i];
  }
  for(int i=0; i<a.length; i++)
  {
    if(a[i] == max)
      count++;
  }
  return count;
}
```

61)

```
int isEvenSubset(int m, int n)
{
  for(int i=2; i<m; i=i+2)
  {
    if(m%i==0 && n%i==0)
      return 0;
  }
  return 1;
}
```

every even factor of m is also a factor of n
if m is an even subset of n

4.

2 even values adjacent to one another

{3, 3, 2, 6, 7}

```
int isTwinoid (int[] a)
{
  for(int i=0; i<a.length; i++)
  {
    if( a[i]%2==0 && a[i+1]%2==1 || a[i]%2==1 && a[i+1]%2==0)
      return 0;
  }
  return 1;
}
```

```
int isTwinoid (int[] a)
{
  for(int i=0; i<a.length; i++)
  {
    if(a[i]%2==0)
      count++;
    if(a[i]%2==0 && a[i+1]%2==0)
      isfoundadj=1;
  }
  if(count==2 && isfoundadj==1)
    return 1;
  else
    return 0;
}
```

43 12

```
int isBalanced(int[] a)
{
  for(int i=0; i<a.length; i++)
    {
      for(int j=0; i<a.length; j++)
        {
          if(a[i] == a[j]*(-1))
              isfound = 1;
        }
        if(isfound == 0)
          return 0;
        }
      return 1;
  }
```

every value n = -n also is in the array

{-2, 3, 2, -3} → 1

{-5, 2, -2} → 0

44

```
int getExponent(int n, int p)
{
  if(p<=1)
    return -1;
  while(n!=0)
    { int r = n%p;
      if(r==0)
        count++;
      n = n/p;
    }
    return count;
}
```

the largest exponent x such that p^x evenly divides n.

(27, 3) → 3 (162, 3) → 4 $2^1 \times 3^4$

(45)

begins with one or more 1's followed by one or more 2's & ends with one or more 1's (begins=ends)

```
int isIdlArray(int[] a)
{
  for(int i=0; i<a.length; i++)
  {
    if(a[i]>2 || a[0]!=1)
      return 0;
  }
  for(int i=0; i<a.length; i++)
  {
    if (a[i] == 1)
        beginCount1 ++;
  }
  for (int i=a.length-1; i>=0; i--)
  {
    if(a[i] == 1)
      endsCount1++;
    }
    if( beginCount1 != endsCount1)
      return 0;
    for (int i=0; a.length; i++)
    {
      if (a[i] == 2)
          isFound2 = 1;
      }
      if ( isFound2 == 0)
        return 0;
      else
      return 1;
}
```

46

{11, 4, 9, 11, 8, 5, 4, 10}

```
int isMaxMinEqual(int a[])
{

    for(int i = 0; i<a.length; i++)
    {
      if (max < a[i])
            max = a[i];
       if(min > a[i])
            min = a[i];
      }
     for(int i=0; i<a.length; i++)
      {
        if(max == a[i])
            maxCount++;

        if(min == a[i])
            minCount++;
        }

        if(maxCount != minCount)
          return 0;

        else
        return 1;

}
```

47/50 evenspaced

largestValue - smallestValue = odd number

```
int isOddSpaced (int[] a)
{
  for(int i = 0 ; i<a.length; i++)
    {
      if (largestValue < a[i])
          largestValue = a[i];
      if (SmallestValue > a[i])
          smallestValue = a[i];
    }
    diff = largestValue - smallestValue;
    if (diff % 2 == 0)
     return 0;
    else
     return 1;
}
```

48 $2<3$ $2+3<6$, $2+3+6<13$

```
int isSuper (int[] a)
{
 for(int i=1 ; i<a.length; i+1)
  {
   for(int j=0; j< i; j++)
    {
     sum = sum + a[i];
    }
    if (sum <= a[i])
      return 0;
  }
 return 1;
}
```

51

* $13 > 2+3+6$, $6 > 2+3$, $3 > 2$

```
int isSub (int[] a)
{
 for(int i = a.length-1; i<=1, i--)
  {
   for(int j = i+1 j<a.length; j++)
    {
     sum = sum + a[i];
    }if (sum >= a[i])
      return 0;
  }
 return 1;
}
```

even/odd Symmetric

even no & odd no upper same order from both directions

49

```
int isSym(int[] a)
{
    int j = a.length-1;
    for(int i=0; i<j; i++, j--)
    {
        if(a[i]%2 != a[j]%2)
            return 0;
    }
    return 1;
}
```

52

```
int hasSingleMode(int[] a)
{
    for(int i=0; i<a.length; i++)
    {
        for(int j=0; j<a.length; j++)
        {
            if(a[i] == a[j])
                count++;
        }
        if(max == count)
            mode++;
        if(max < count)
            max = count;
            mode = 1;
    }
        if(mode == 1)
            return 1;
        else
            return 0;
}
```

most frequently appearing value.
single mode

countOnes(9) return 2 because the binary representation

```
2 | 9 - 1
2 | 4 - 0
2 | 2 - 0
    1
```

```
int countOnes (int n)
{
    while (n != 0)
    {
        int r = n%10
        n = n/10
        if (r == 1)
            count++

    }
    return count;
}
```

cube-perfect

```
int isCubePerfect (int [] a)
{
    for (int i = 0; i < a.length; i++)
    {
        if (i*i*i == a[i])
            return 1;
    }
    return 0;
}
```

page-43

```
int isZeroLimited (int[] a)
{
    int seq = 1
    for(int i=0; i<a.length; i++)
        {
            if(i == seq && a[i] != 0)
              return 0;
            if( i != seq && a[i] == 0)
              return 0;
        }
      return 1;
}
```

page-43

```
int iszeroBalanced2(int[] a)
{
    for(int i=0; i<a.length; i++)
      {
        for(int j=0; j<a.length; j++)
          {
            if( a[i] == a[j]*(-1))
                isfound = 1;
          }
          if (isfound == 0)
              return 0;
      }
      return 1;
}
```

page-43

{1, 0, 5, -1, 0, 2, 3}

i	2*n+1	seq
1	a[2*0+1] == 0	
4	1	1
	2	5
	3	7
		10

page-47

{1, 2, -2, -1} = 1

```
int iszeroBalanced(int[] a)
{
  for(int i=0; i<a.length; i++)
    {
      sum = sum + a[i];
      if (sum == 0)
        return 1;
    }
  return 0;
}
```

4. (b)

```
int largestAdjacentSum(int[] a)
{
  int largestSum = a[0] + a[1];
  for(int i=0; i<a.length; i++)
  {
      if(largestSum < a[i] + a[i+1])
          largestSum = a[i] + a[i+1];
  }
  return largestSum;
}
```

```
int isConsecutiveFactors(int n)
{
  for(int i=1; i<n/2; i++)
  {
    if(n%i==0 && n%(i+1)==0)
      return 1;
  }
  return 0;
}
```

j = i+1

3 = 2+1

```
void updateMileage(int[] a, int miles)
{
    for(int i = a.length-1; i>=0; i--)
    {
        num = num*10 + a[i];
    }
    num = num + miles
    int index = 0;
    while (num != 0)
    {
        int dig = num%10;
        num = num/10;
        a[index] = dig;
        index++;
    }
}
```

Q7

```
        if(a[i]==a[i+1]
            count++;
        if(count<2)
            return 0;
            count=2;
        }
    return 1;
}
Double eval(double x, int[] a)
{
    for(int i=0; i<a.length; i++)
    {
        sum = sum + a[i]*pow(x,i)
    }
    return sum;
}
int pow(int a, int i)
{
    for(int j=0; j<i; j++)
    {
        power = power * a;
    }
    return power;
```