Assignment - ①

1. Draw flowchart that checks two numbers and print maximum of two numbers.

→

```
        ┌─────────┐
        │  start  │
        └────┬────┘
             │
        ┌────▼──────┐
        │ Enter a,b │
        └────┬──────┘
             │
   True      ▼
 ┌───────   ◇ a>b ◇
 │         
 ▼
┌─────────┐
│ print a │
└────┬────┘
     │       │ false
     │       ▼
     │   ┌─────────┐
     │   │ print b │
     │   └────┬────┘
     │        │
     │   ┌────▼────┐
     └──►│   end   │
         └─────────┘
```
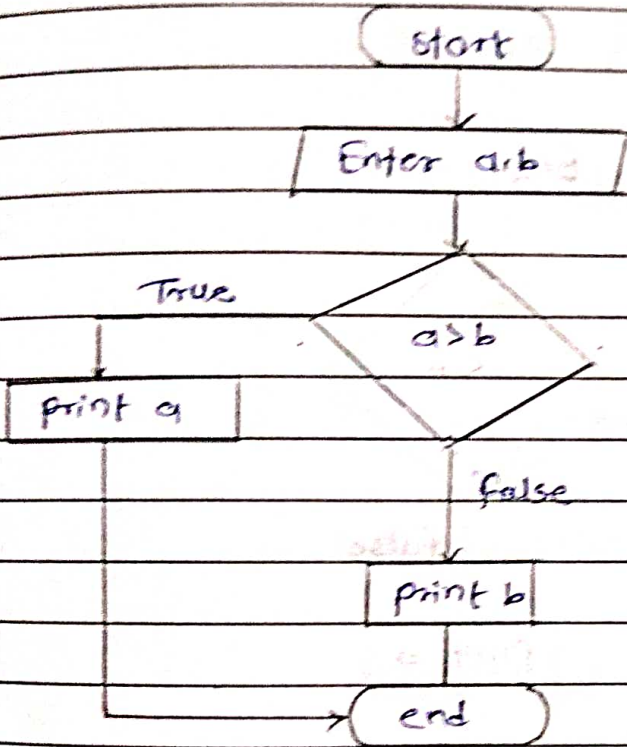
2. Draw flowchart that checks two Numbers and print minimum of two Numbers.

$\rightarrow$

```
                    ┌──────────┐
                    │  Start   │
                    └────┬─────┘
                         │
                 ┌───────▼────────┐
                 │   Enter a,b    │
                 └───────┬────────┘
                         │
                         ▼
         True          ╱ ╲
      ┌──────────────◄ a≤b ►
      │              ╲ ╱
      ▼               │ False
 ┌─────────┐          ▼
 │ print a │    ┌──────────┐
 └────┬────┘    │ print b  │
      │         └────┬─────┘
      │              ▼
      │        ┌──────────┐
      └───────►│   End    │
               └──────────┘
```
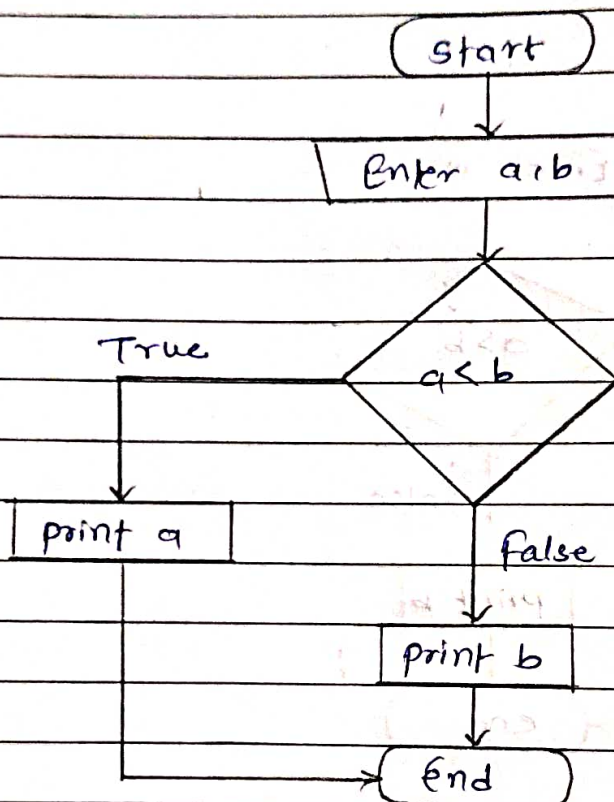
3. Draw flowchart that reads number in inches and converts it to meters.

```
        ( Start )
            |
            v
  /------------------\
 /  Enter Number in   \
 \  inches (a)         \
  \------------------/
            |
            v
  +------------------------+
  |  a = 0.0254 meters     |
  |  res = a × 0.0254 m.   |
  +------------------------+
            | T
            v
  +------------------+
  | print result     |
  +------------------+
            |
            v
        ( end )
```
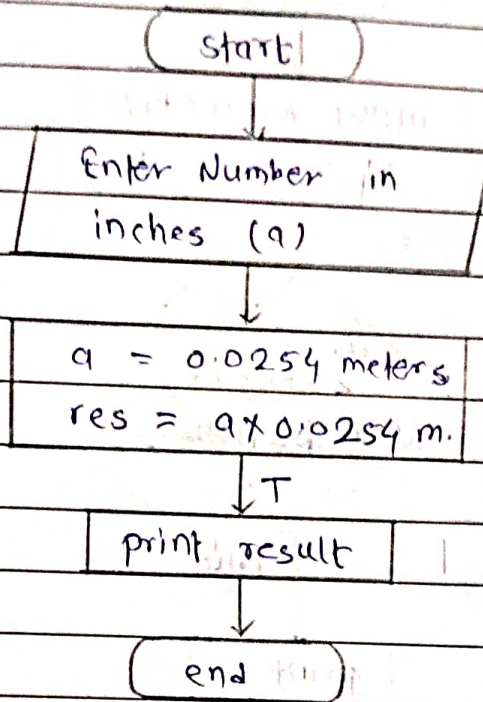
4. Draw flowchart to print 2,4,6,8,10

→

```
                    ┌─────────────┐
                    │    Start    │
                    └─────────────┘
                           │
                           ▼
                   ╱─────────────────╲
                  │  input Num (N)    │
                  │  2,4,6,8,10       │
                   ╲─────────────────╱
                           │
                           ▼
                      ╱─────────╲
                     ╱    if.    ╲
                    │  2<N>10     │
                     ╲ &&  N=N+2 ╱
                      ╲─────────╱
                           │ True
                           ▼
                    ┌─────────────┐
                    │   print N   │
                    └─────────────┘
                           │
                           ▼
                    ┌─────────────┐
                    │     end     │
                    └─────────────┘
```
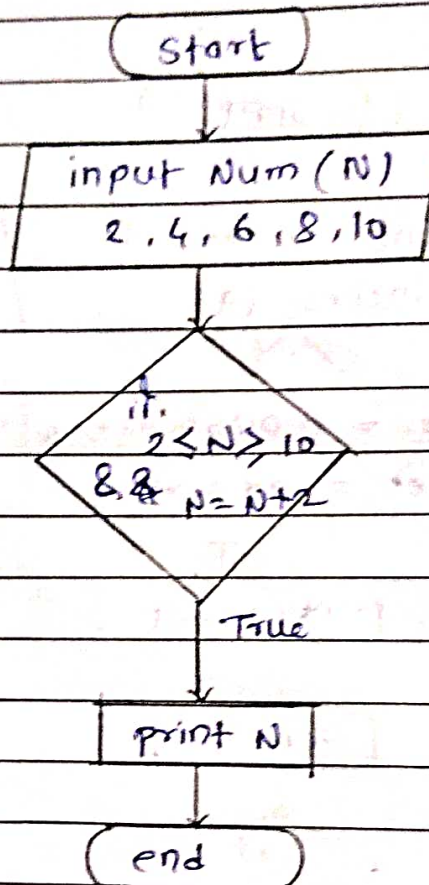
Assignment - ②

1. Write algorithm that prints table of any num.

→   i.e. 7

```
        ┌──────────┐
        │  start   │
        └──────────┘
             │
    ┌─────────────────────┐
    │ initialize Num = 7  │
    └─────────────────────┘
             │
   ┌──────────────────────────┐
   │  set  table = Num * i    │
   └──────────────────────────┘
             │
          ╱  for  ╲
        ╱ i=1; i≥10; ╲
        ╲    i++    ╱
          ╲       ╱
             │ T
    ┌──────────────┐
    │  print table │
    └──────────────┘
             │
        ┌──────────┐
        │   end    │
        └──────────┘
```
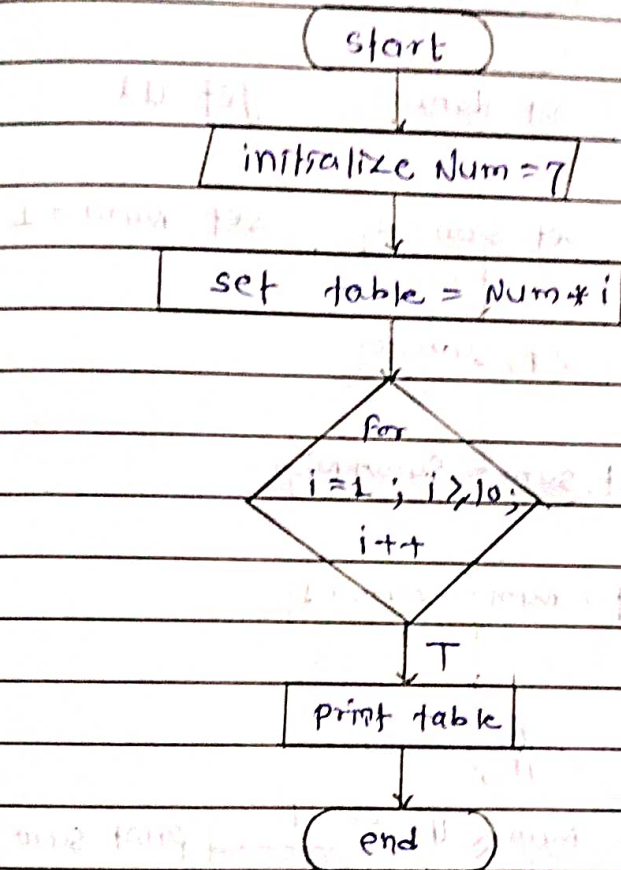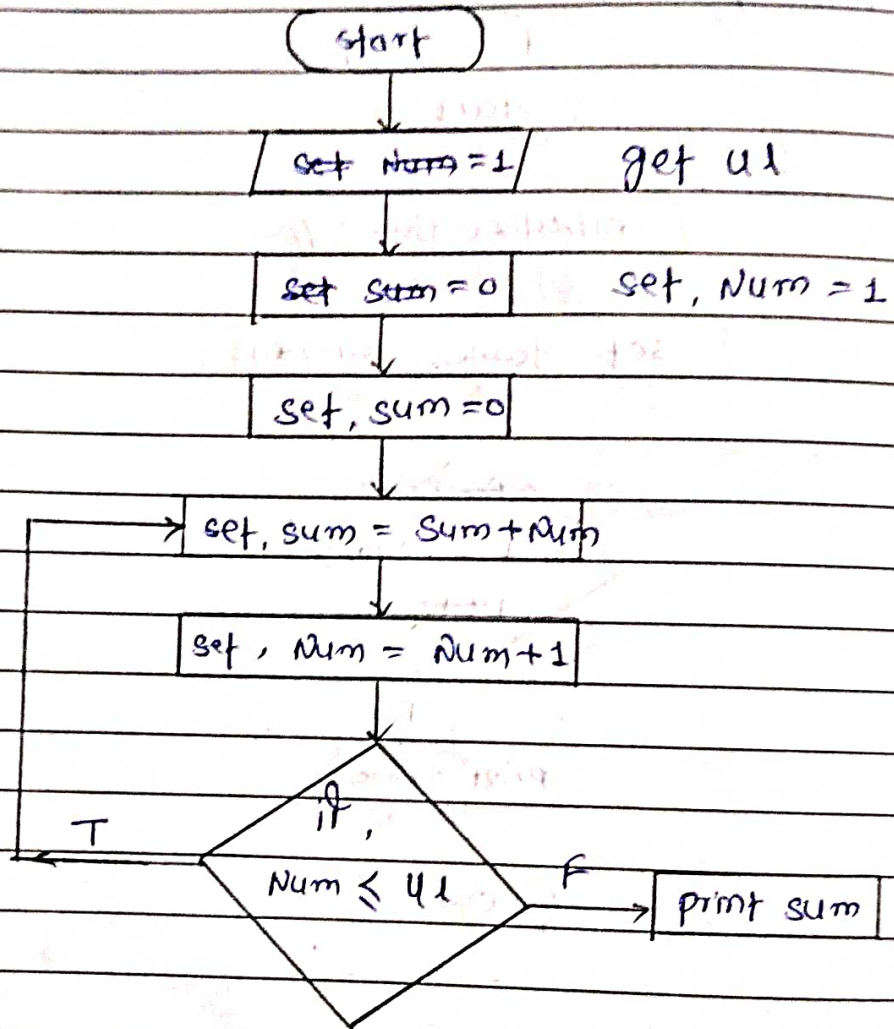
algorithm

- initialize num = 7
- set table = Num * i
- for , (i = 1 ; i ≥ 10 , i++)
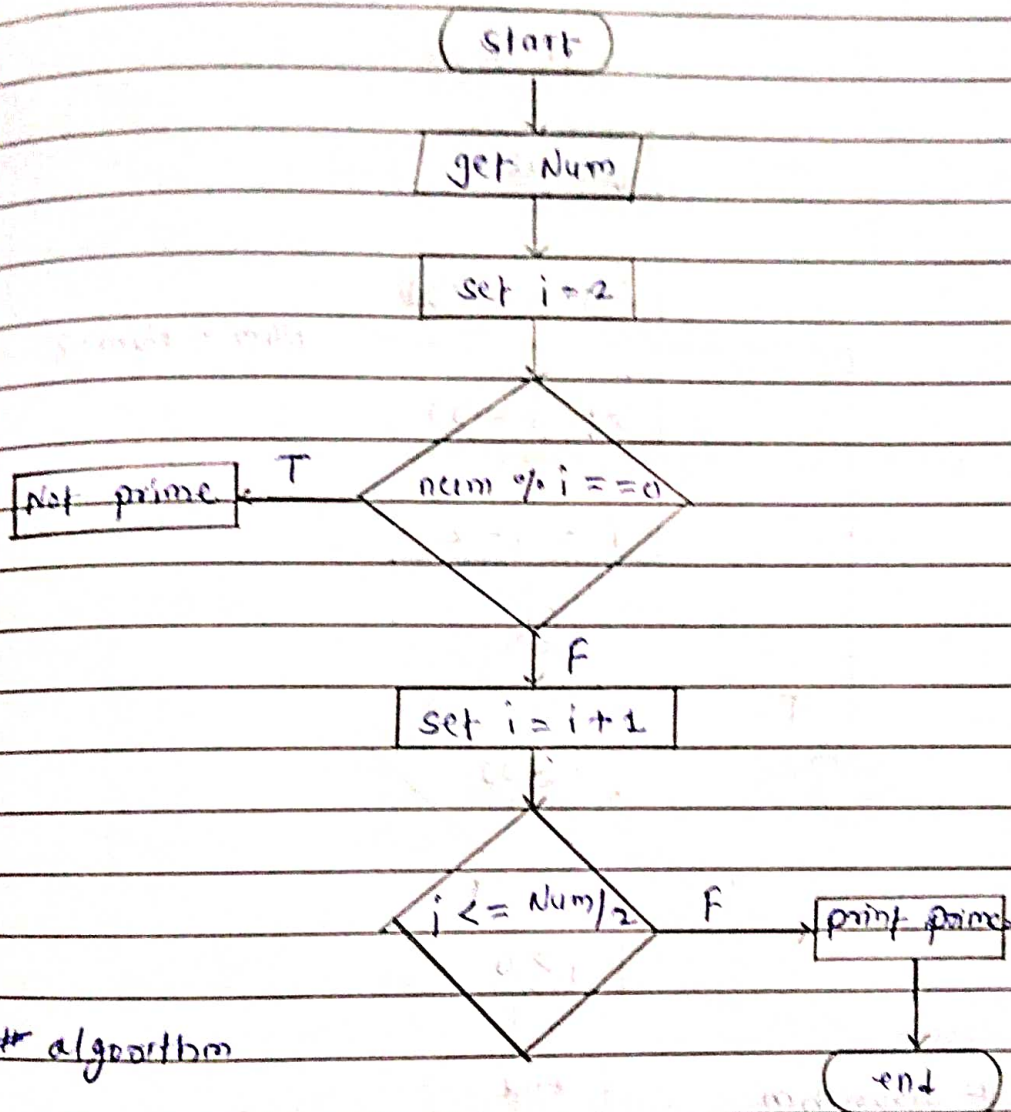   print table
- end

2. Write algorithm that print sum of N Numb...

$\longrightarrow$

```
        ( Start )
            |
            v
   [ Set Num = 1 ]        get ul
            |
            v
   [ Set Sum = 0 ]        set, Num = 1
            |
            v
   [ set, sum = 0 ]
            |
            v
 ->[ set, sum = Sum + Num ]
 |          |
 |          v
 |  [ set, Num = Num + 1 ]
 |          |
 |          v
 |        / if, \
 |   T  /        \  F
 <-----<  Num ≤ ul  >----->[ print sum ]
        \        /
          \    /
```

\# algorithm

- initialize upper limit, ul
- set, Num = 1
- set, sum = 0
- set, sum = sum + Num
- set, Num = Num + 1
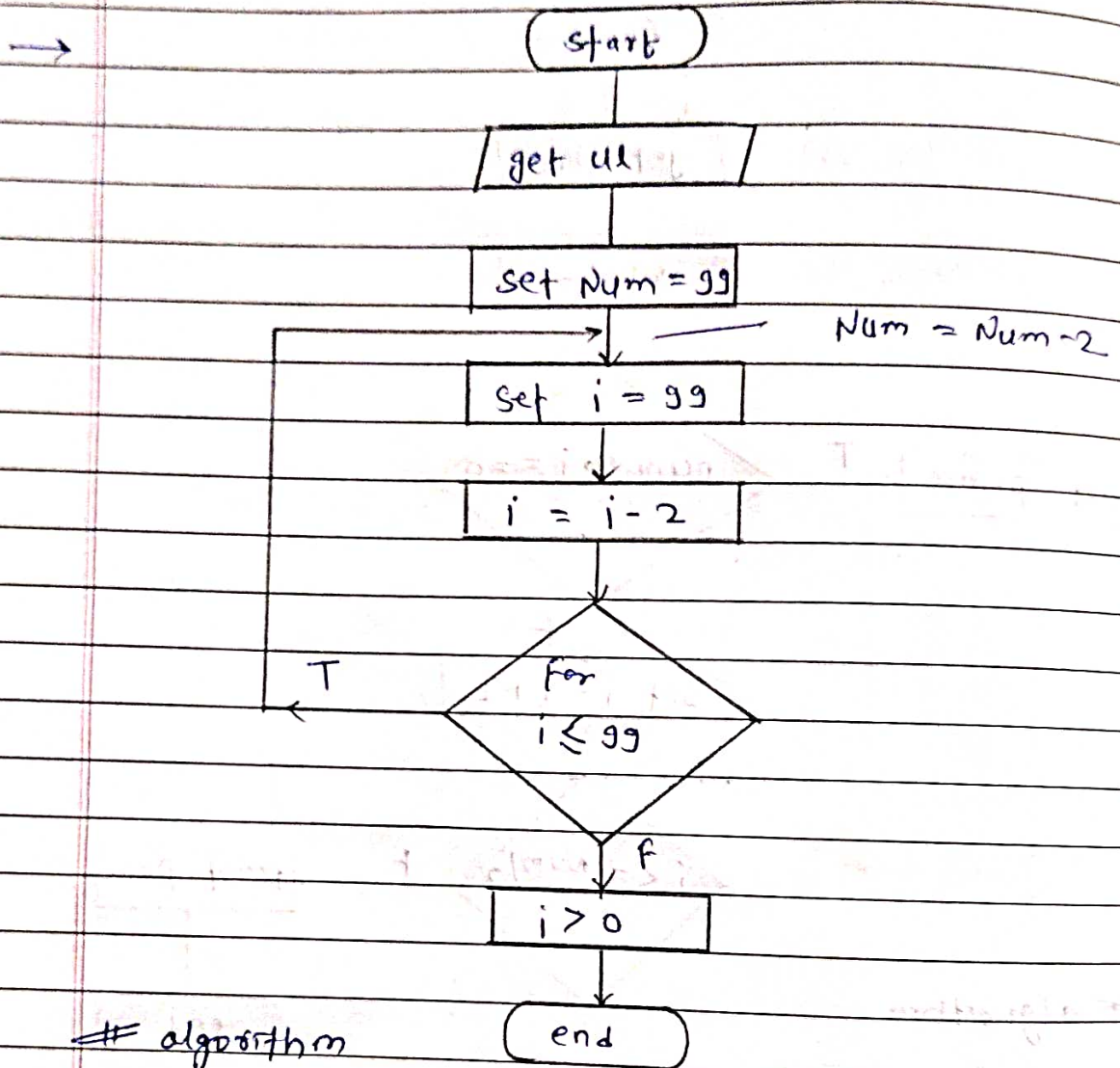- if, (Num ≤ ul)
  print sum
- end.

1. Write algorithm that to check if a number is prime.

→

```
              ( Start )
                  |
             [ get Num ]
                  |
             [ set i=2 ]
                  |
                  ▼
 [Not prime]◄─T─ < num % i == 0 >
                  |
                  F
                  |
             [ set i = i+1 ]
                  |
                  ▼
            < i <= Num/2 > ─F─► [ print prime ]
                                       |
                                       ▼
                                   ( end )
```

algorithm

- initialize number
- set , i = 2
- check if, num % i == 0
- set , i = i+1
- check if , i <= Num/2
  false then , print "Prime Number"
- end.

4. Write algorithm to print all odd numbers backward from 99 to 1.

```
        ( start )
            |
        / get ul /
            |
      | set Num = 99 |
            |
            |------------------  Num = Num-2
            ↓
      | set i = 99 |
            |
      | i = i - 2 |
            |
           / \
          /   \
     T   / for \
    <---|       |
          \ i≤99 /
           \   /
            \ /
             | f
             ↓
      | i > 0 |
            |
            ↓
        ( end )
```

algorithm

- initialize upper limit, ul
- set Num = 99
- set Num = Num - 2
- set i = 99
- set i = i - 2
- for, (i ≤ 99)
  print Num'
- check i > 0 and stop

4. Write algorithm to print all odd numbers backward from 99 to 1.

```
                    ( start )
                        |
                  / get ul /
                        |
                 | set Num = 99 |
                        |                    Num = Num-2
         +------------> |         ———
         |         | Set  i = 99 |
         |              |
         |         | i = i - 2 |
         |              |
         |            /    \
      T  |           / for  \
         +----------<  i ≤ 99 >
                      \     /
                       \   /
                         | f
                         |
                    |  i > 0  |
                         |
# algorithm          ( end )
```
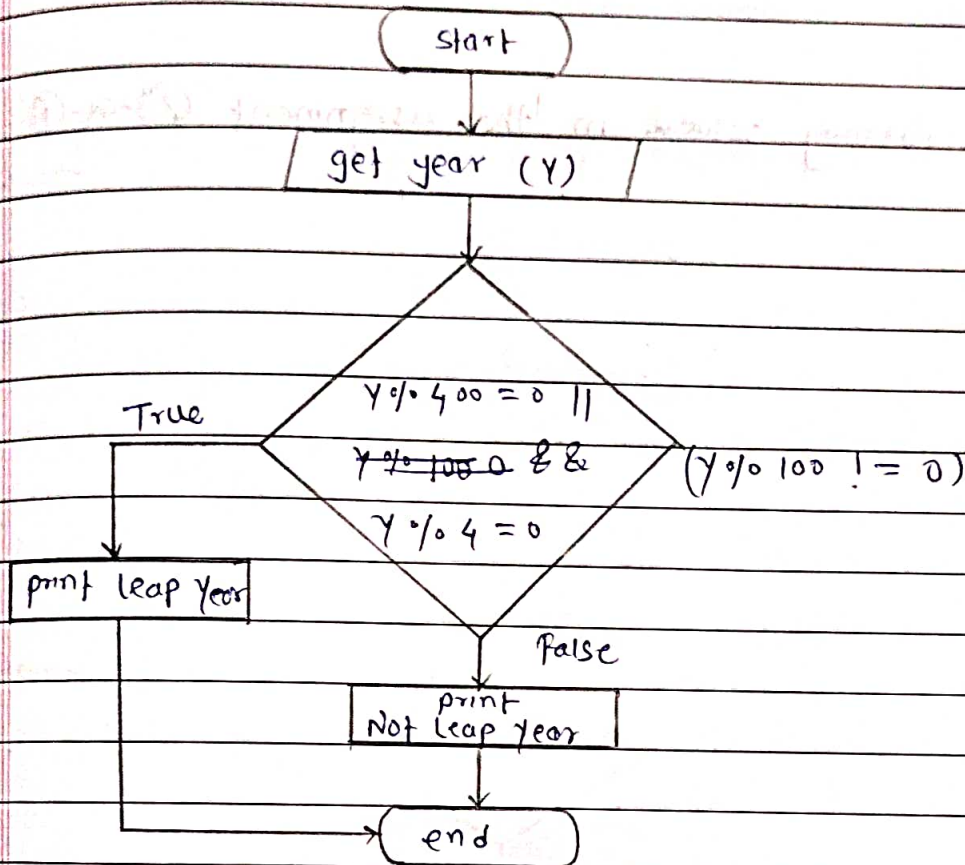
- initialize upper lmt , ul
- set Num = 99
- set Num = Num - 2
- set i = 99
- set i = i-2
- for , (i ≤ 99)
- print Num
- check i > 0 and stop

Assignment - ③

1. check if a year is leap or not.

→

```
                    ╭─────────╮
                    │  start  │
                    ╰─────────╯
                         │
                         ▼
              ┌────────────────────┐
              │   get year (Y)     │
              └────────────────────┘
                         │
                         ▼
                       ╱   ╲
      True           ╱       ╲
   ┌────────────────◆  Y % 400 = 0 ||  ◆──── (Y % 100 != 0)
   │                 ╲  Y % 100 0 &&  ╱
   ▼                  ╲  Y % 4 = 0  ╱
┌──────────────┐       ╲          ╱
│print leap year│        ╲       ╱
└──────────────┘            │ False
   │                        ▼
   │            ┌────────────────────┐
   │            │      print         │
   │            │   Not leap year    │
   │            └────────────────────┘
   │                        │
   │                        ▼
   │                   ╭─────────╮
   └──────────────────▶│   end   │
                       ╰─────────╯
```

algorithm

- initialize year (Y)
- if (Y % 4 = 0) or (Y % 400 = 0)
  and   (y % 100 != 0)
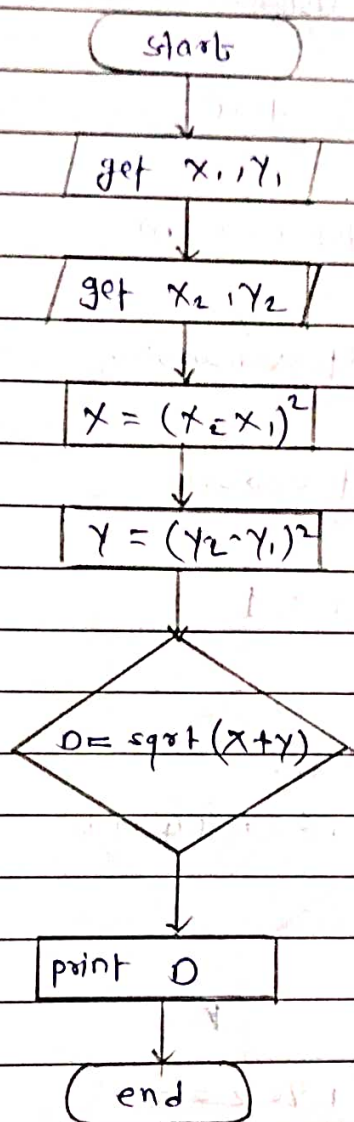- is true print leap year
- if not then print Not leap year
- stop

2. Write algorithm to print all odd numbers backward from 99 to 1.

→

Already solved in the assignment (2) - Q. ④

3. Java program to calculate distance between two points

$\rightarrow$

```
start
   |
get $x_1, y_1$
   |
get $x_2, y_2$
   |
$x = (x_2 - x_1)^2$
   |
$y = (y_2 - y_1)^2$
   |
$D = sqrt(x+y)$
   |
print D
   |
end
```
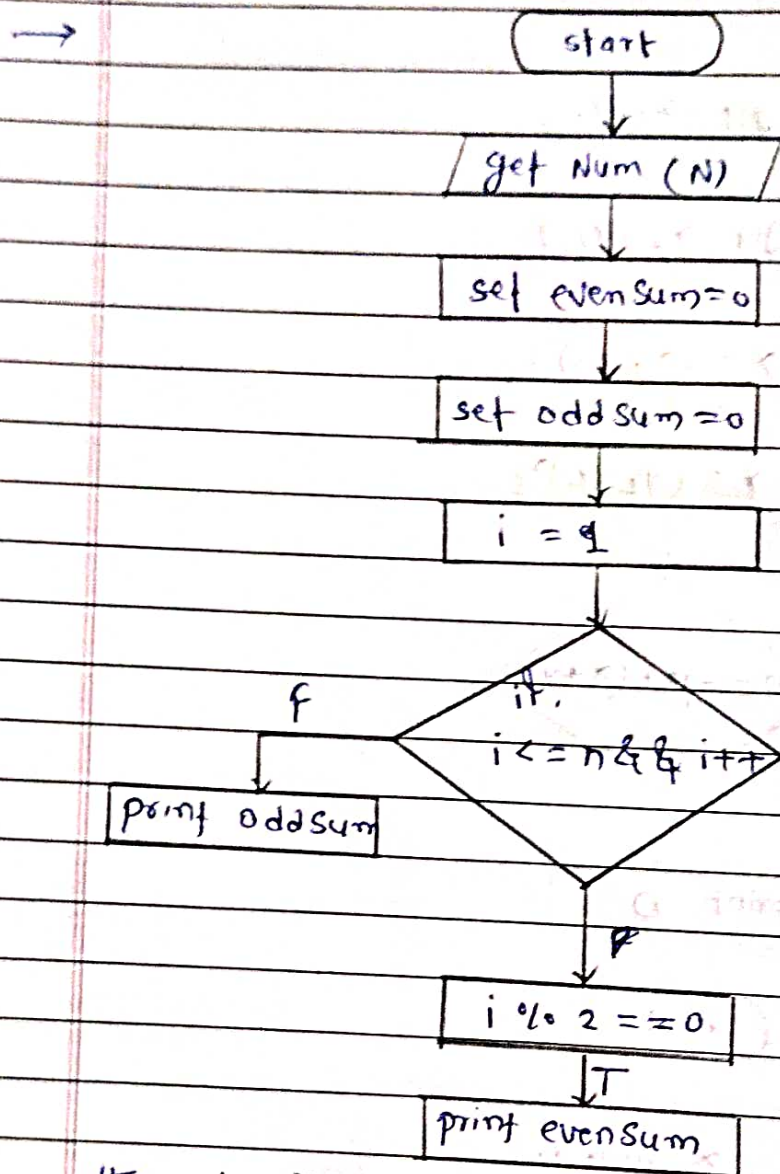
algorithm

- initialize x_1 and y_1
- initialize x_2 and y_2
- set $x = (x_2 - x_1)^2$
- set $y = (y_2 - y_1)^2$
- set D = sqare root of $(x+y)$
- print D
- stop.

4. Write algorithm to print sum of even and odd digits, considering 10 Numbers are taken from user.

```
                    ┌─────────┐
              ───→  │  start  │
                    └─────────┘
                         │
                         ▼
                  ╱ get Num (N) ╱
                         │
                         ▼
                 ┌──────────────┐
                 │ set evenSum=0│
                 └──────────────┘
                         │
                         ▼
                 ┌──────────────┐
                 │ set oddSum=0 │
                 └──────────────┘
                         │
                         ▼
                 ┌──────────────┐
                 │    i = 1     │
                 └──────────────┘
                         │
                         ▼
                       ╱╲
         f           ╱    ╲  it.
      ┌──────────────   i<=n&&i++  ╲
      │           ╲          ╱
      ▼             ╲      ╱
┌──────────────┐      ╲  ╱
│ print oddSum │       ╲╱
└──────────────┘        │ F
                        ▼
                 ┌──────────────┐
                 │  i % 2 == 0  │
                 └──────────────┘
                        │ T
                        ▼
                 ┌──────────────┐
                 │ print evenSum│
                 └──────────────┘
```

— if false, print oddsum.

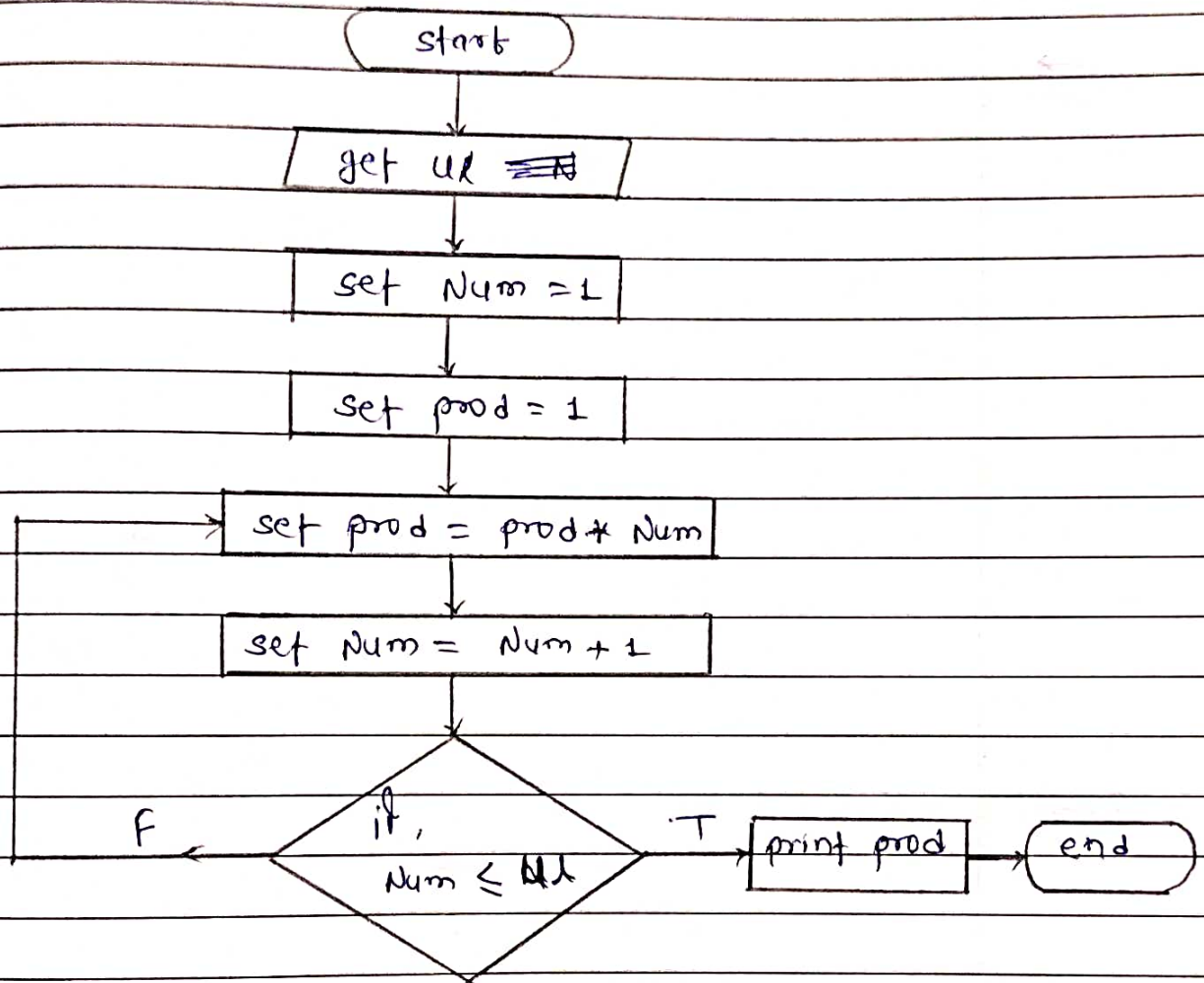\# algorithm
   - initialize Number, N
   - set evenSum = 0
   - set oddSum = 0
   - set i = 0
   - check if (i <= n) and (i++)
      if True check (i % 2 == 0)
      and print even sum

5. Calculate product of (sum) digits of Number. (till N)

→

```
                    ( start )
                        |
                        v
            +-----------------------+
            |   get ul = N          |
            +-----------------------+
                        |
                        v
            +-----------------------+
            |   set Num = 1         |
            +-----------------------+
                        |
                        v
            +-----------------------+
            |   set prod = 1        |
            +-----------------------+
                        |
      +---------------> +-----------------------+
      |                 | set prod = prod * Num |
      |                 +-----------------------+
      |                        |
      |                        v
      |          +-----------------------+
      |          | set Num = Num + 1     |
      |          +-----------------------+
      |                        |
      |                        v
      |                   /\
      |                  /  \
   F  |      F         /  if, \     T      +------------+      ( end )
      +<--------------<  Num <= ul  >----->| print prod |----->
                        \        /          +------------+
                         \      /
                          \    /
                           \  /
                            \/
```

algorithm

- initialize upper limit = N (ul)
- set Num = 1
- set Num = 1
         prod
- set prod = prod * Num
- set Num = Num + 1
- if .(Num < ul)
    True then print 'prod'
- stop.