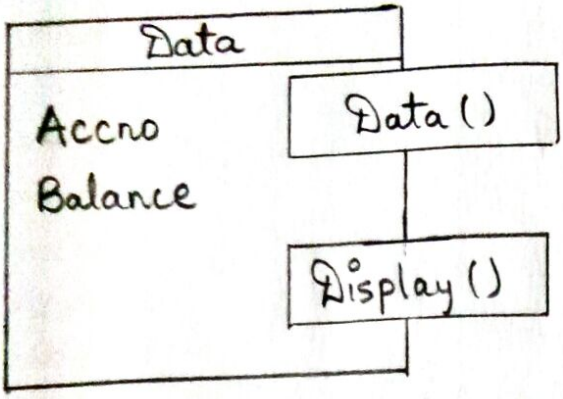
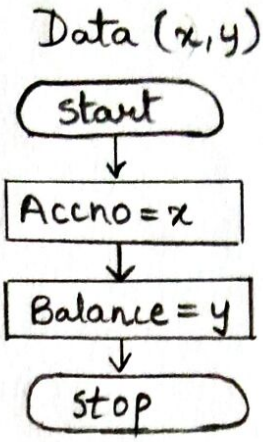
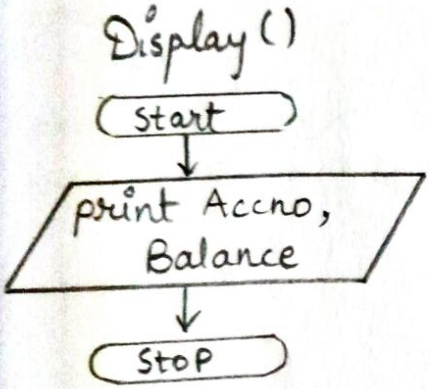
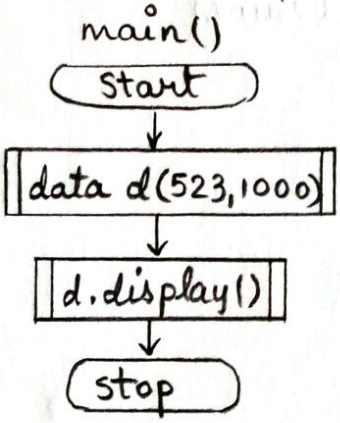
// Program 1

Flowchart :-









* Algorithm :-

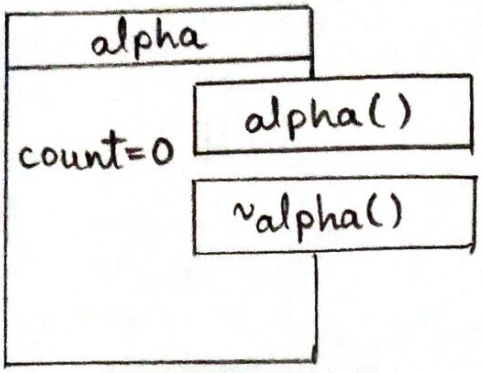
data() ⇒ ① Start
② Accno = x
③ Balance = y
④ Stop

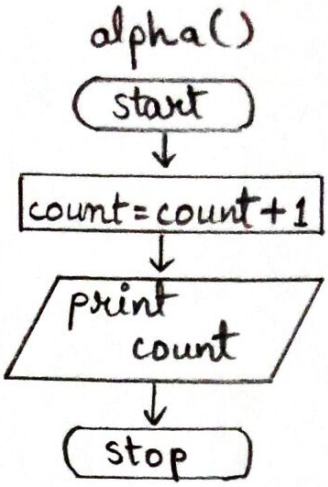
display() ⇒ ① Start
② Print Accno and Balance
③ Stop.

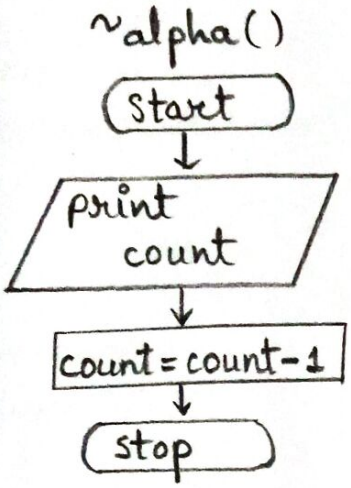
main() ⇒ ① Start
② data d(523, 1000)
③ Call d.display()
④ Stop

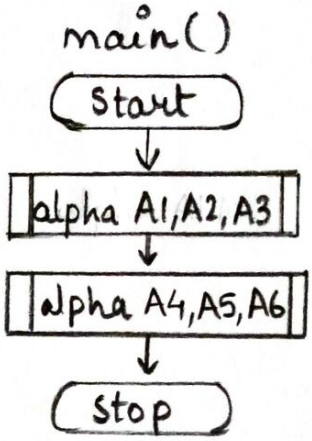
Program 2

* Flowchart :-









Algorithm :-

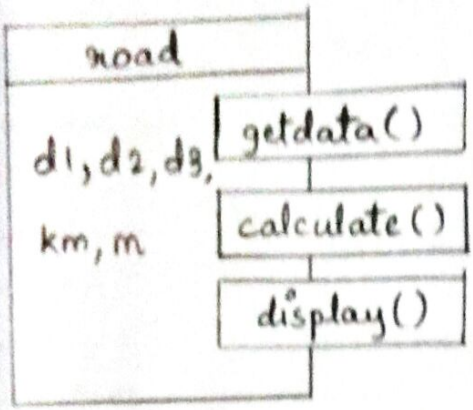
alpha() ⇒
1. start
2. count = count + 1
3. print count
4. stop

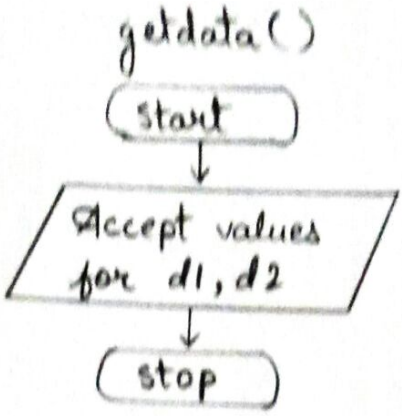
~alpha() ⇒
1. start
2. print count
3. count = count - 1
4. stop

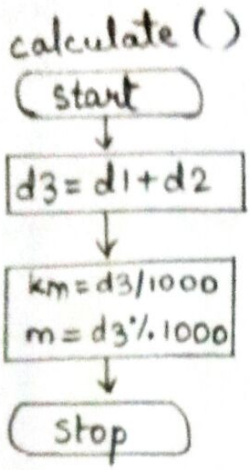
main() ⇒
1. start
2. alpha A1, A2, A3
3. alpha A4, A5, A6
4. stop

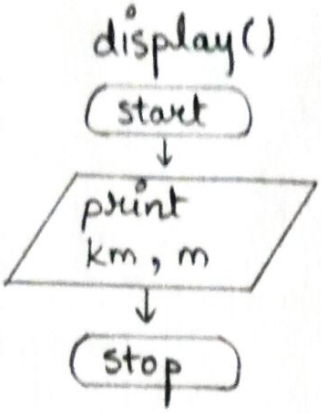
// Program 3

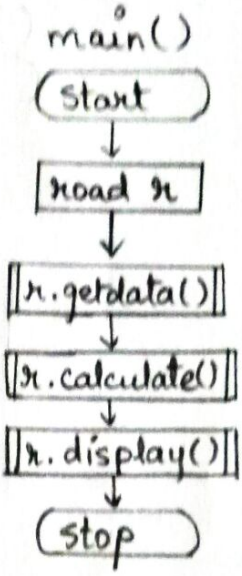
* Flowchart :-











* Algorithms :-

getdata() ⇒
1. Start
2. Accept values in d1, d2
3. Stop

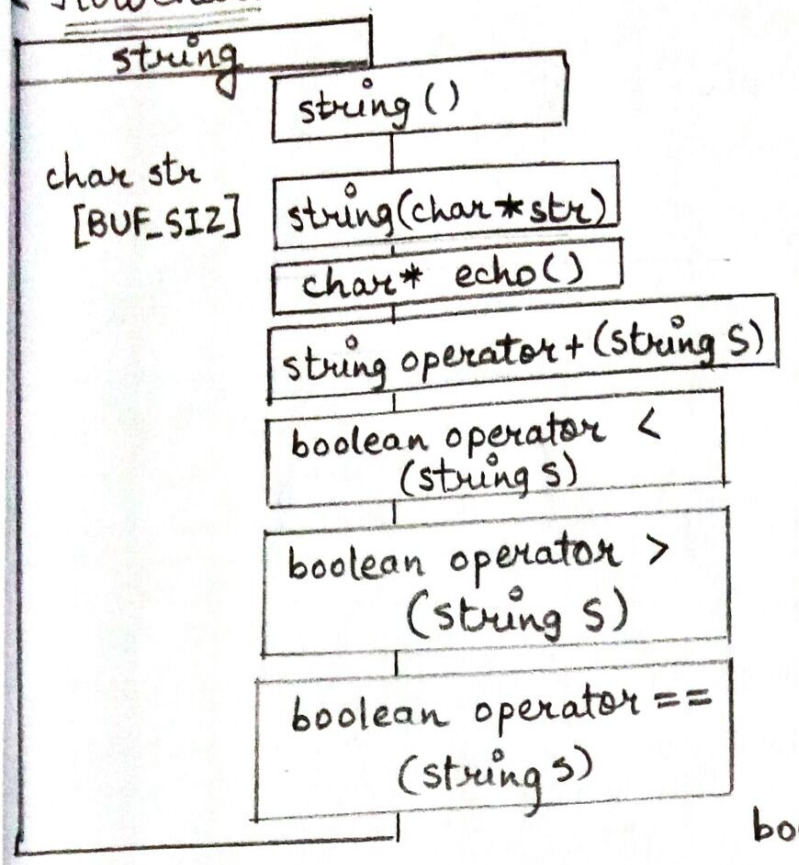
calculate() ⇒
1. start
2. Perform d3 = d1+d2
3. Perform km = d3/1000 and m = d3%1000
4. stop

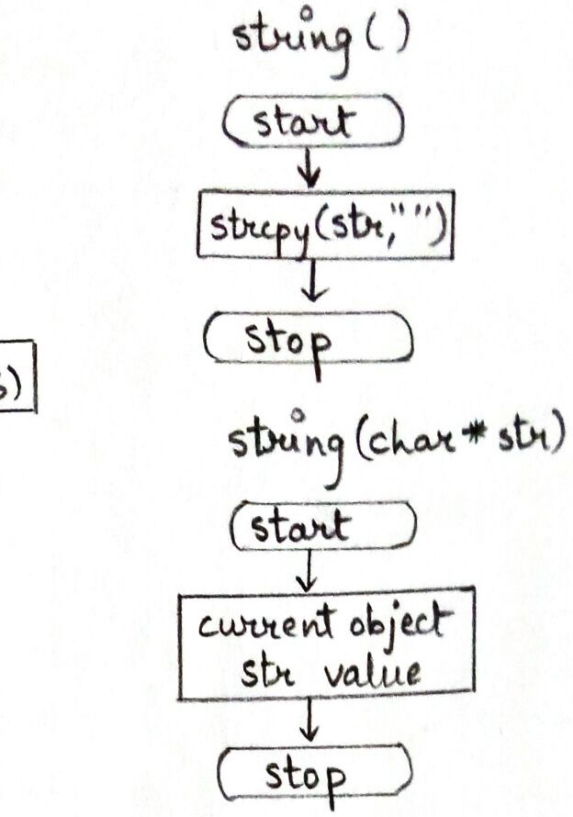
display() ⇒
1. start
2. Print km, m
3. stop

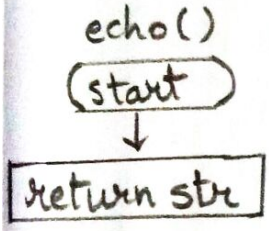
main() ⇒
1. start
2. read r.
3. Call r.getdata()
4. Call r.calculate()
5. Call r.display()
6. stop

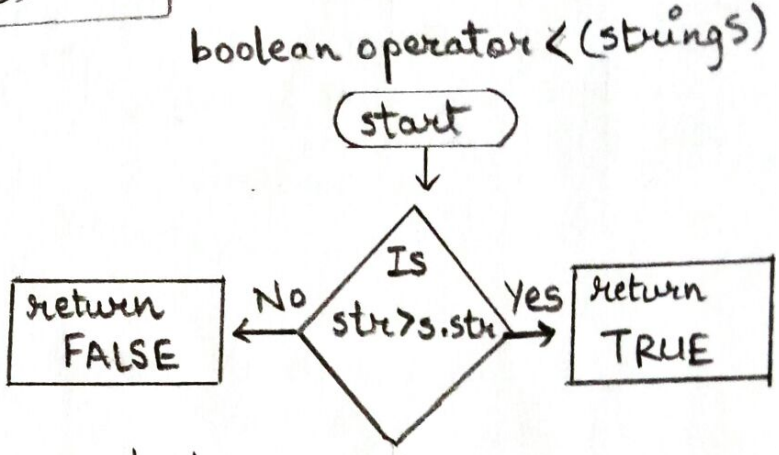
// Program 4

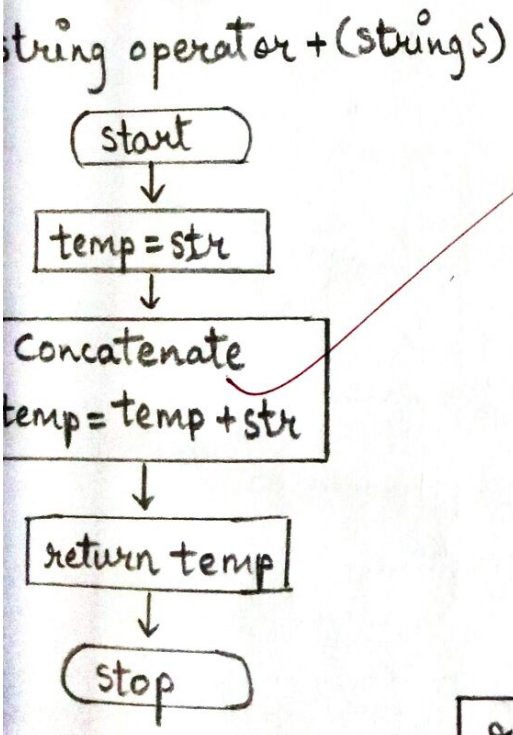
Flowchart:-

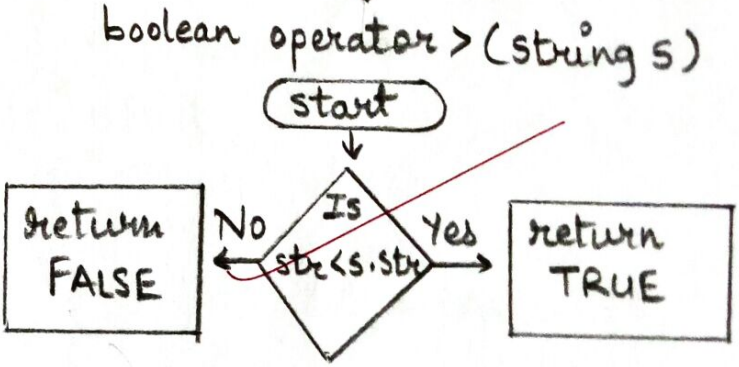


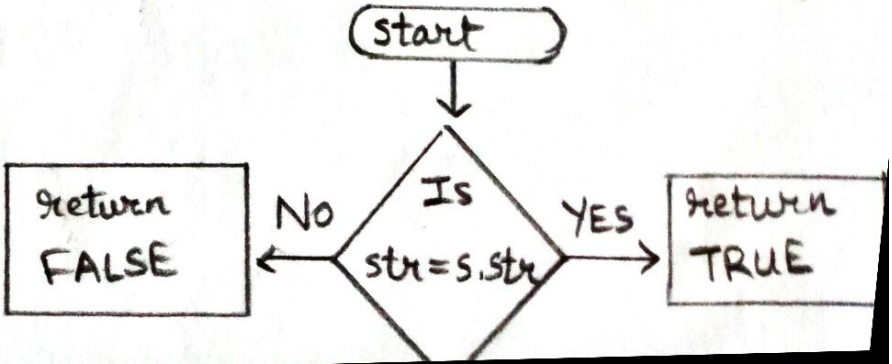


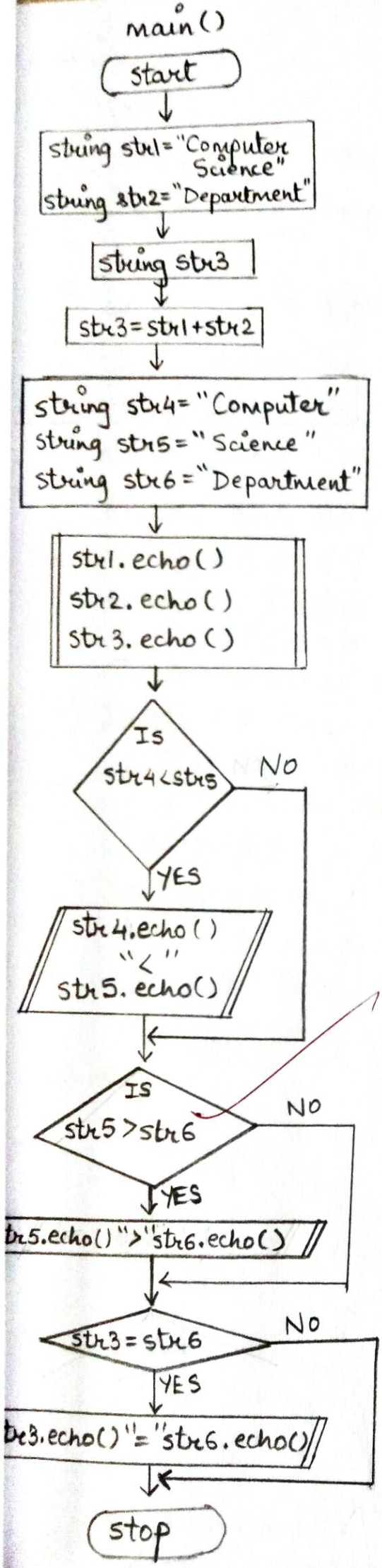












* Algorithm:-

- string()

1. start
2. strcpy(str," ")
3. stop

- string(char * str)

1. start
2. current object str value
3. stop

- string operator +

1. start
2. temp = str
3. temp = temp + str.
4. return temp

- operator <

1. start
2. If str > s.str
   - YES: return TRUE
   - NO: return FALSE

- operator >

1. start
2. If str < s.str
   - YES: return TRUE
   - NO: return FALSE

- operator ==

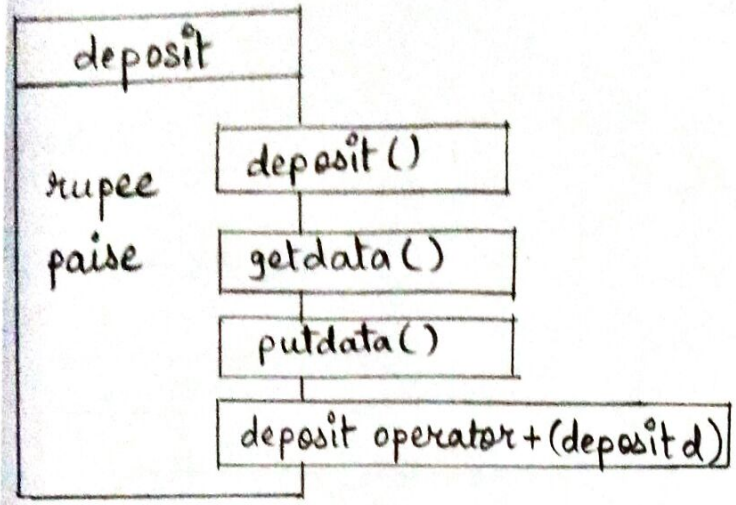
1. start
2. If str == s.str
   - YES: return TRUE
   - NO: return FALSE

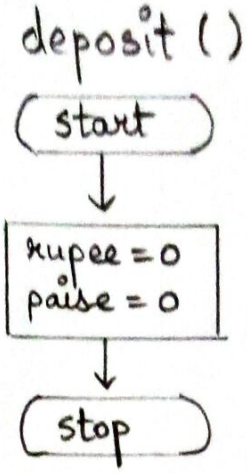
- main()

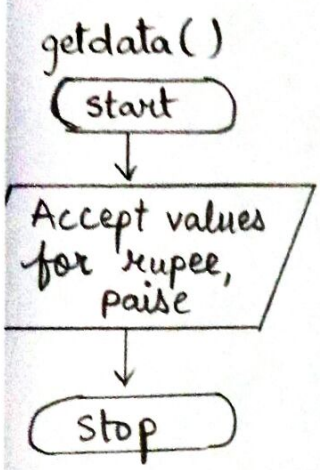
① start
② string str1 = "Computer Science"
③ string str2 = "Department"
④ string str3.
⑤ str3 = str1 + str2.
⑥ string str4 = "Computer"
⑦ string str5 = "Science"
⑧ string str6 = "Department"
⑨ Display str1, str2, str3 using echo() function.
⑩ Is str4 < str5 ?
  YES: (a) Display str4.echo() "<" str5.echo()
       (b) Jump to step 11.
  NO: (a) Jump to step 11.
⑪ Is str5 > str6
  YES: (a) Display str5.echo() ">" str6.echo().
       (b) Jump to step 12.
  NO: (a) Jump to step 12.
⑫ Is str3 = str6
  YES: (a) Display str3.echo() "=" str6.echo().
       (b) Jump to step 13.
  NO: (a) Jump to step 13.
⑬ stop.

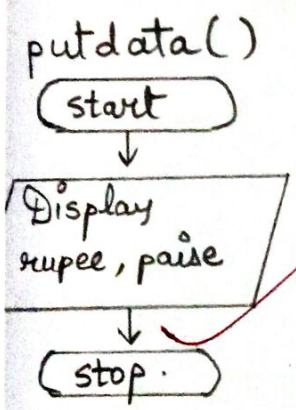
Program 5 :

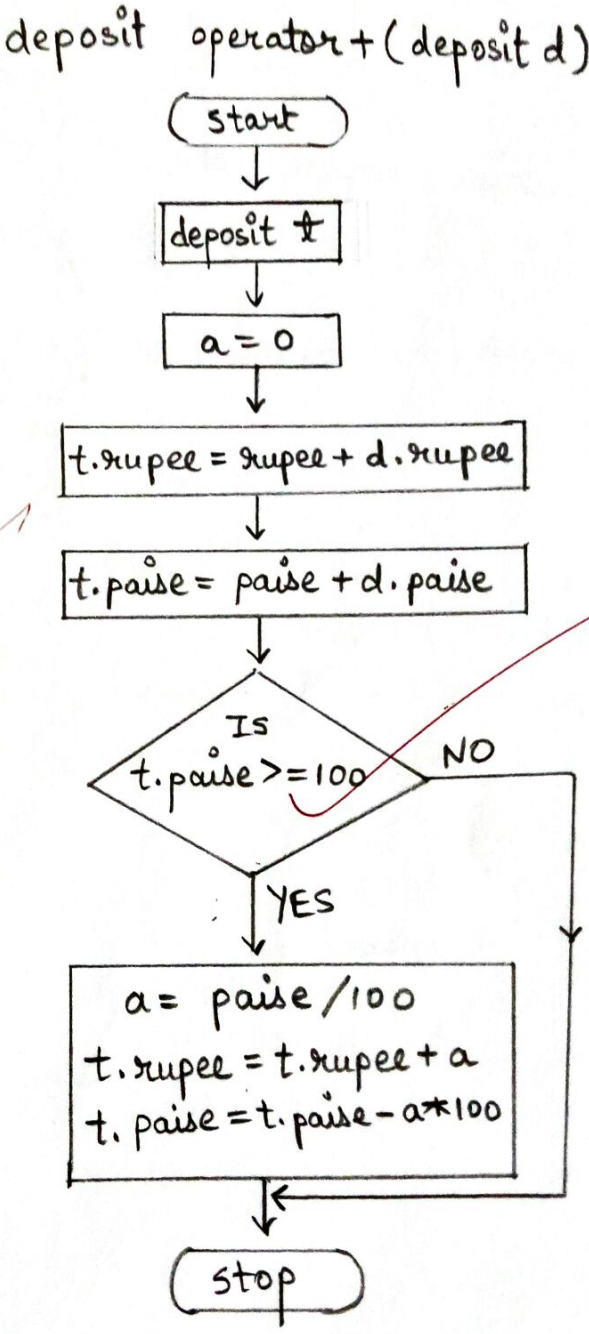
* Flowchart

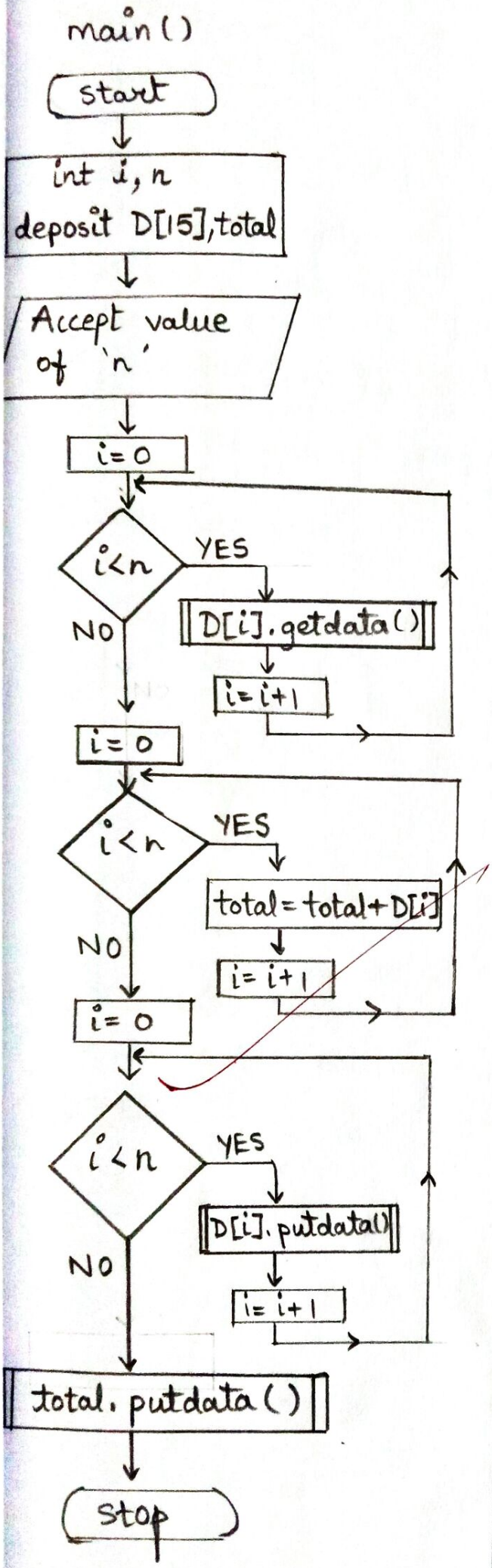












*Algorithm

- deposit()

① Start
② rupee = 0
③ paise = 0
④ stop

- getdata()

① start
② Accept values for rupee, paise
③ stop

- putdata()

① start
② Display rupee, paise
③ stop.

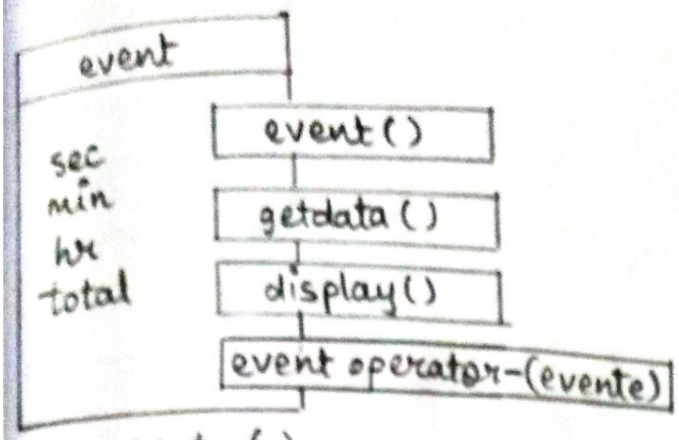
- deposit operator+(deposit d)

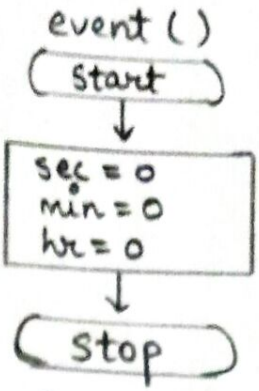
① start
② a = 0
③ deposit t
④ t.rupee = rupee + d.rupee
⑤ t.paise = paise + d.paise
⑥ if t.paise >= 100
  YES: (a) a = paise/100
    (b) t.rupee = t.rupee + a
    (c) t.paise = t.paise - a*100
    (d) Jump to step 7
  NO: (a) Jump to step 7
⑦ Stop

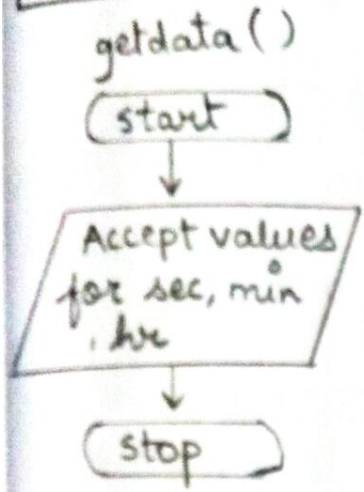
• main()

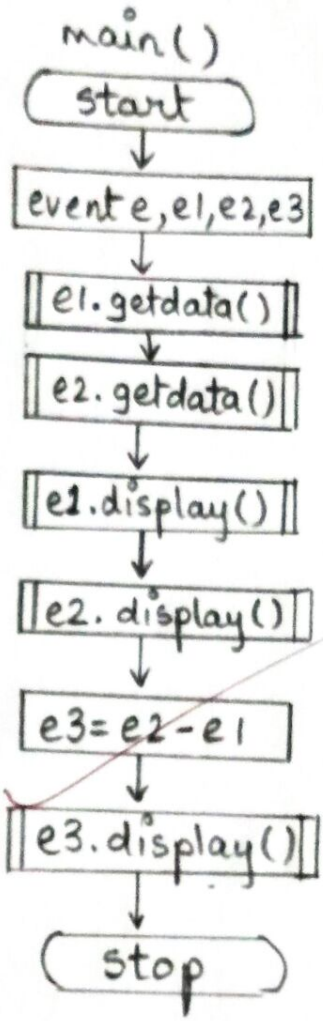
① Start
② int i, n
③ deposit D[15], total
④ Accept value for 'n'
⑤ Initialize i=0
⑥ Is i<n?
   YES: (a) Call D[i].getdata()
        (b) i=i+1
        (c) Jump to step 6.
   NO: (a) Jump to step 7.
⑦ Initialize i=0
⑧ Is i<n?
   YES: (a) total=total+D[i]
        (b) i=i+1
        (c) Jump to step 8
   NO: (a) Jump to step 9
⑨ Initialize i=0
⑩ Is i<n?
   YES: (a) Call D[i].putdata()
        (b) i=i+1
        (c) Jump to step 10
   NO: (a) Jump to step 11
⑪ Call total.putdata()
⑫ Stop

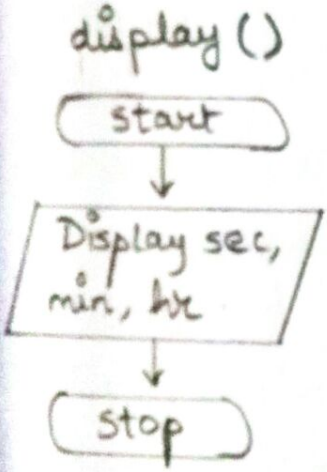
// program 6

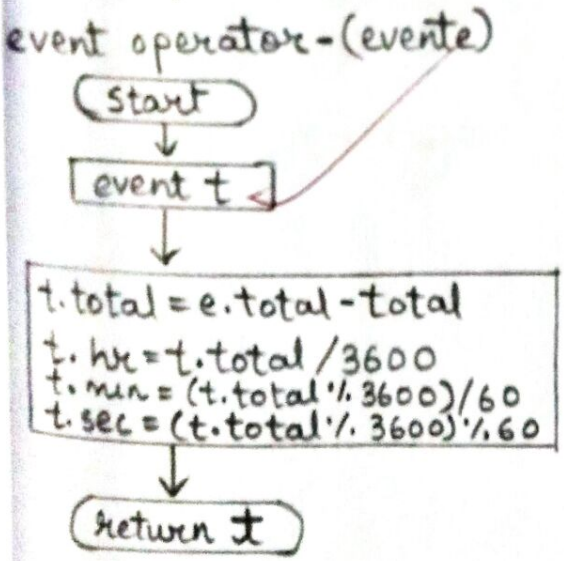

event
sec
min
hr
total
event()
getdata()
display()
event operator-(evente)


event()
start
sec = 0
min = 0
hr = 0
stop


getdata()
start
Accept values for sec, min, hr
stop


main()
start
event e,e1,e2,e3
e1.getdata()
e2.getdata()
e1.display()
e2.display()
e3=e2-e1
e3.display()
stop


display()
start
Display sec, min, hr
stop


event operator-(evente)
start
event t
t.total = e.total - total
t.hr = t.total / 3600
t.min = (t.total % 3600)/60
t.sec = (t.total % 3600) % 60
return t

* Algorithm

• event ()

① start
② sec = 0, min = 0, hr = 0
③ stop

• getdata ()

① start
② Accept values for hr, min, sec
③ Stop

• putdata ()

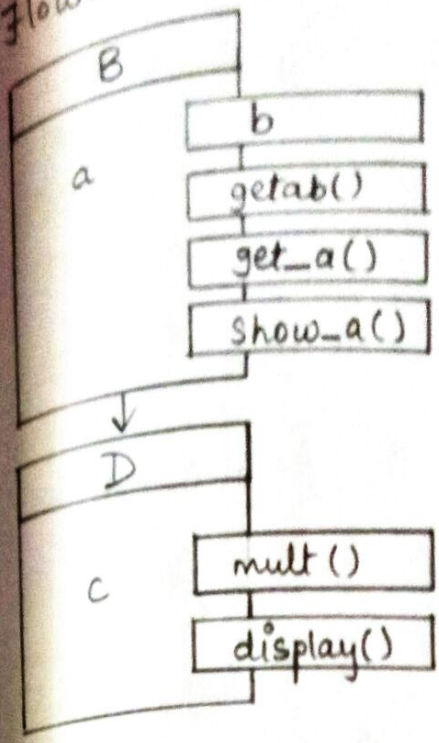
① start
② Display hr, min, sec
③ stop.

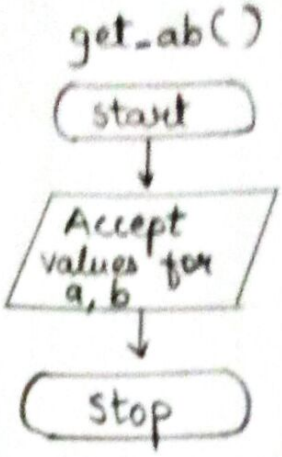
• main ()

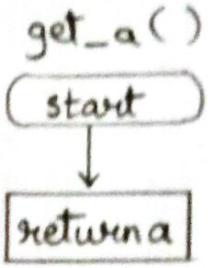
① start
② event e, e1, e2, e3
③ Call e1. getdata ()
④ Call e2. getdata ()
⑤ Call e1. display()
⑥ Call e2. display()
⑦ e3 = e2 + e1
⑧ Call e3. display ()
⑨ stop.

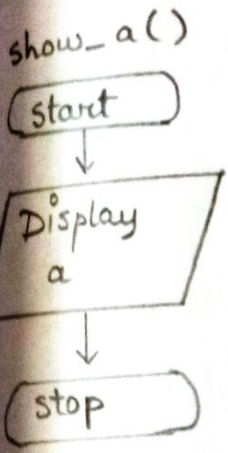
// Program 7

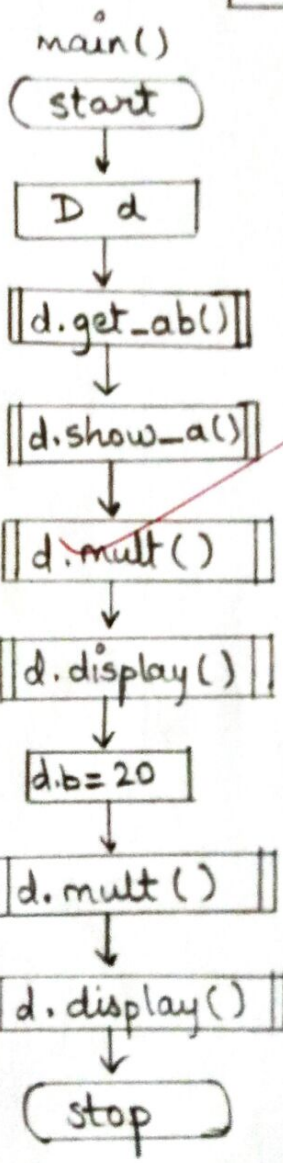
* Flowchart :-

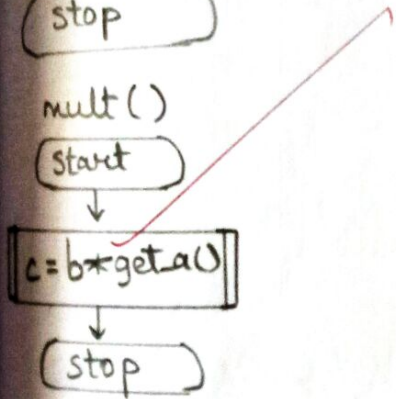


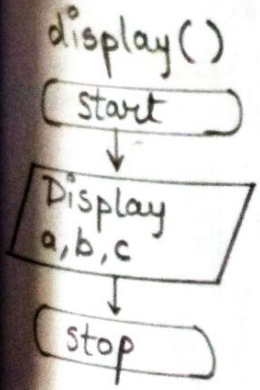












* Algorithms

• get_ab()
① start
② Accept values for a, b.
③ stop

• get_a()
① start
② return a

• show_a()
① start
② Display a
③ stop

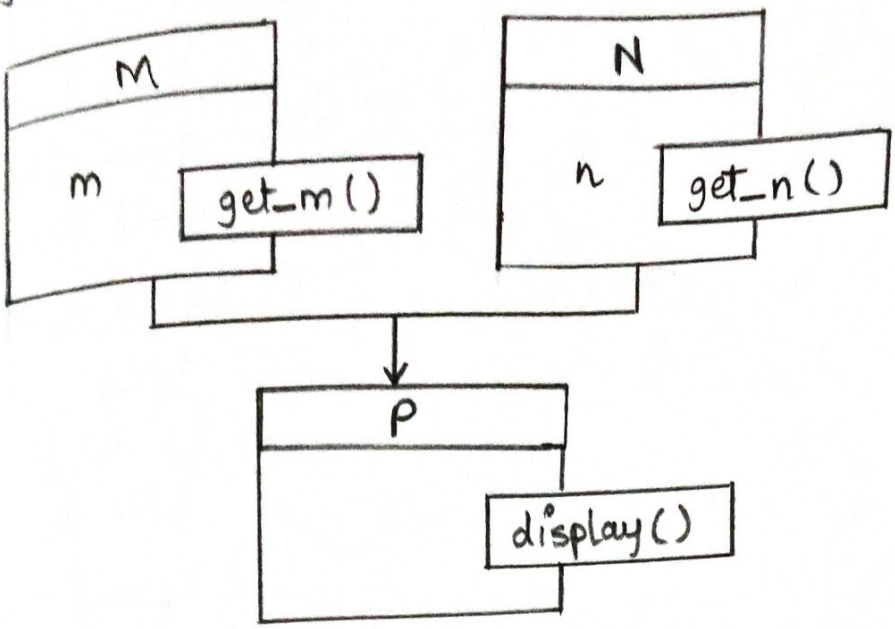
• mult()
① start
② c = b * get_a()
③ stop.

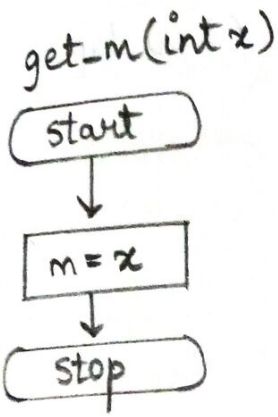
• display()
① start
② Display a, b, c
③ Stop.

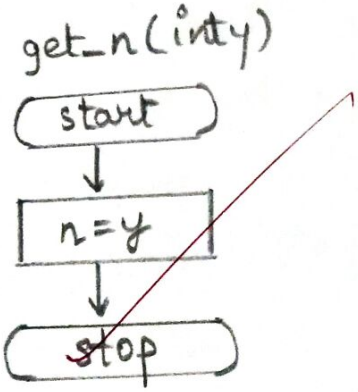
• main()
① start
② D d
③ Call d.get_ab()
④ Call d.show_a()
⑤ Call d.mult()
⑥ Call d.display()
⑦ Assign d.b = 20
⑧ Call d.mult()
⑨ Call d.display()
⑩ stop

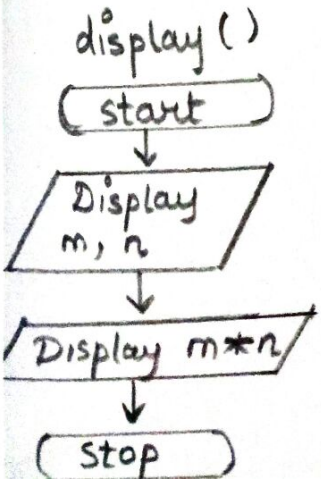
// Program 8

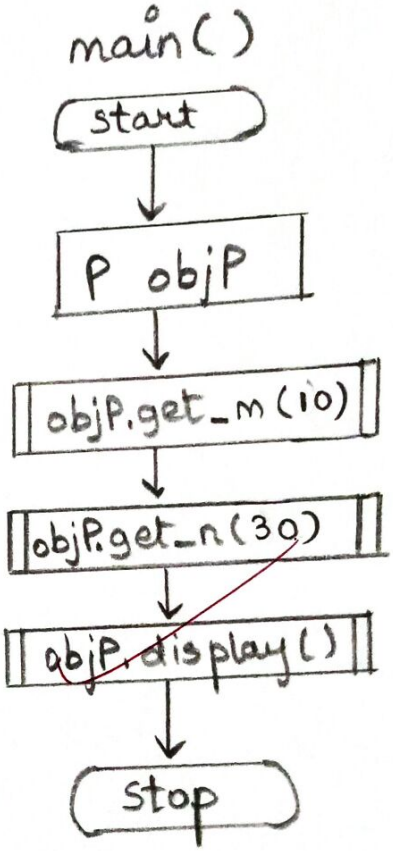
* Flowchart :









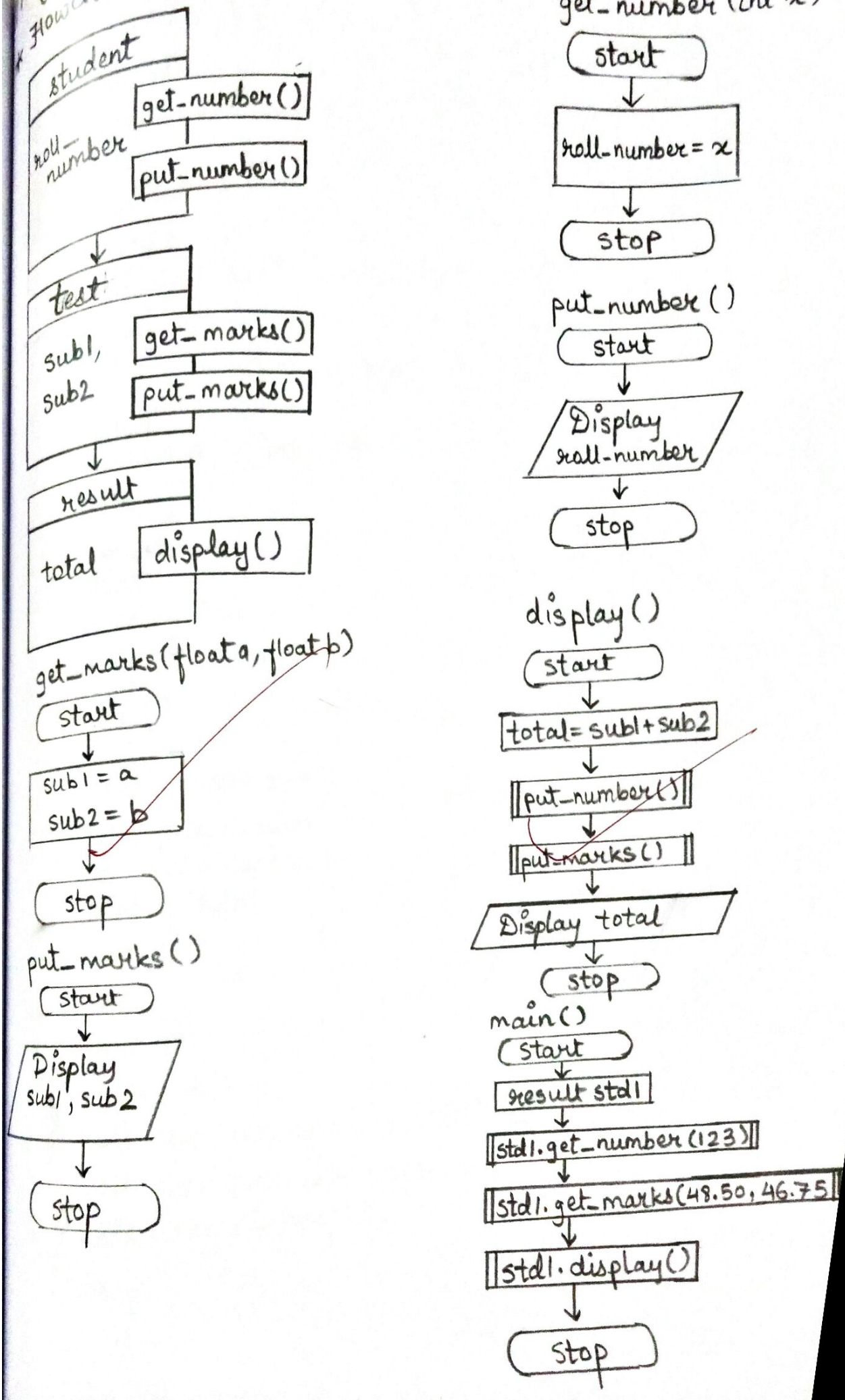


* Algorithm:

- get_m (int x)
  1. start
  2. m = x
  3. stop
- get_n (int y)
  1. start
  2. n = y
  3. stop
- display ()
  1. start
  2. Display m, n
  3. Display m * n
  4. Stop
- main ()
  1. start
  2. P objP
  3. Call get_m (10)
  4. Call get_n (30)
  5. Call display ()
  6. Stop

Program 9

* Flowchart:



* Algorithm

• get_number (int x)
① start
② roll_number = x
③ stop

• put_number ()
① start
② Display roll_number
③ stop

• get_marks (float a, float b)
① start
② sub1 = a , Sub2 = b
③ stop

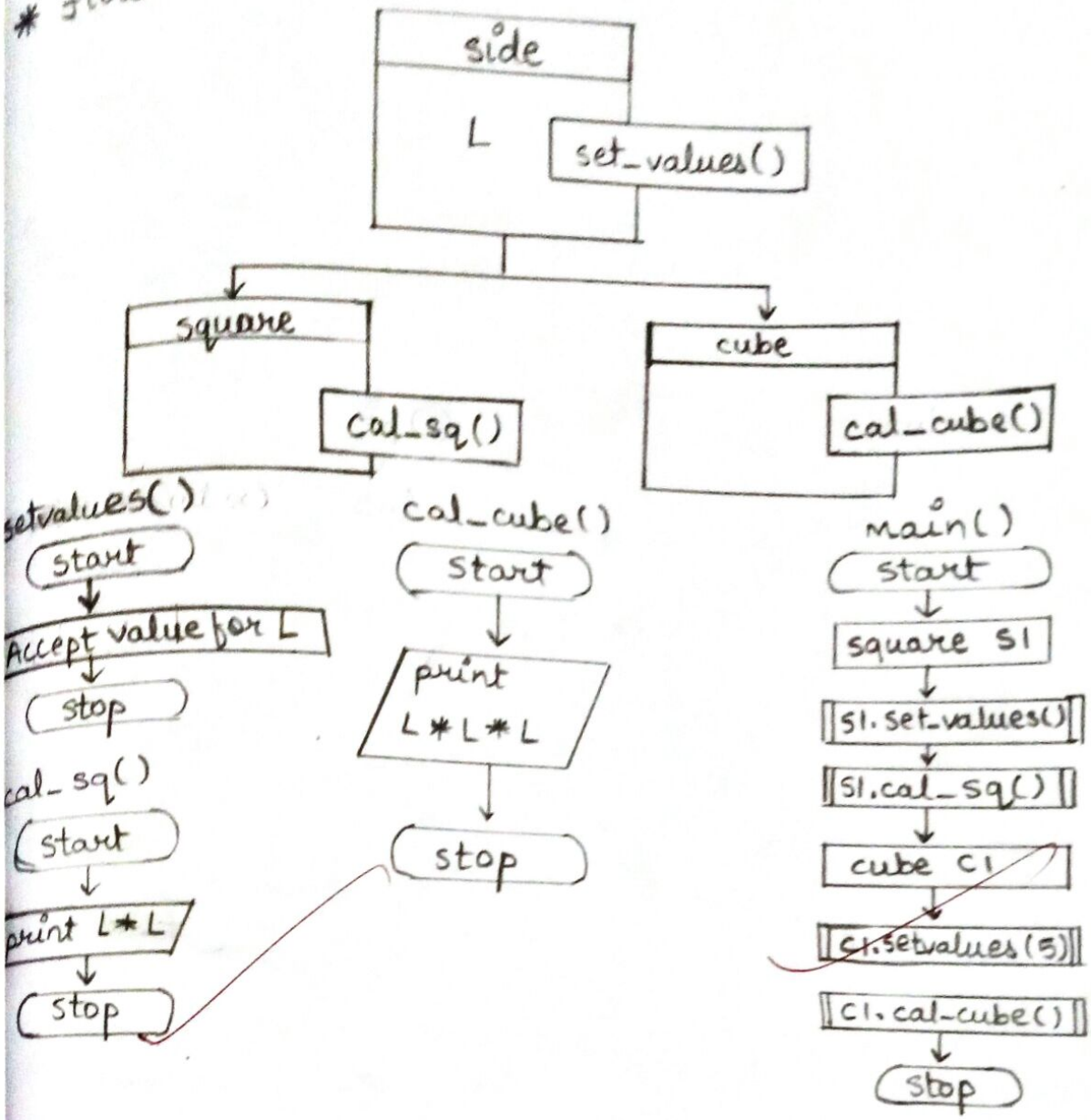
• put_marks ()
① start
② Display Sub1 , Sub2
③ stop

• display ()
① start
② total = sub1 + sub2
③ Call put_number ()
④ Call put_marks ()
⑤ Display Total
⑥ Stop

• main ()
① start
② result std1
③ Call std1. get_number (123)
④ Call std1. get_marks (48.50, 46.75)
⑤ Call std1. display ()
⑥ Stop

// program 10

* Flowchart :-



• Algorithm:

• set_values()

① start
② Assign value for L
③ stop

cal_sq()

① start
② Print L*L
③ stop

cal_cube()

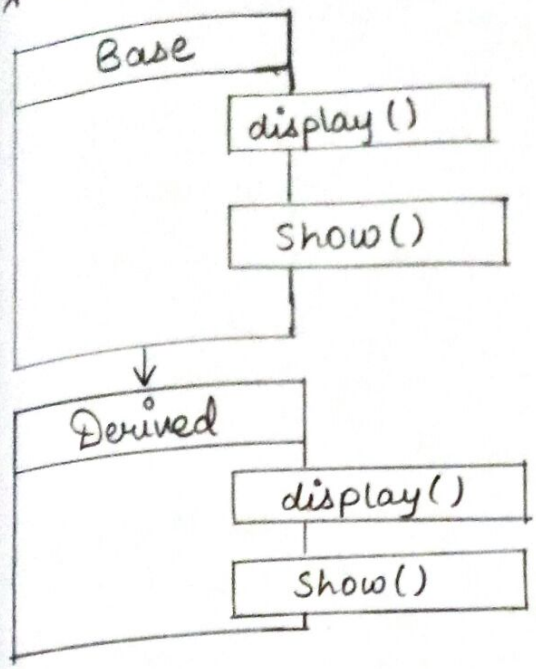
① start
② Print L*L*L
③ stop

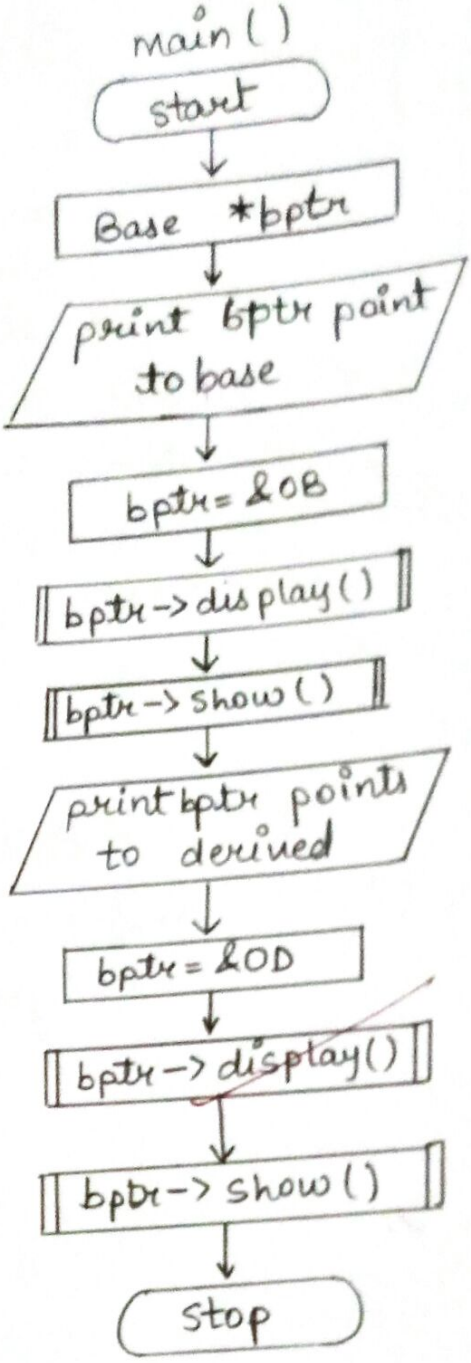
• main()

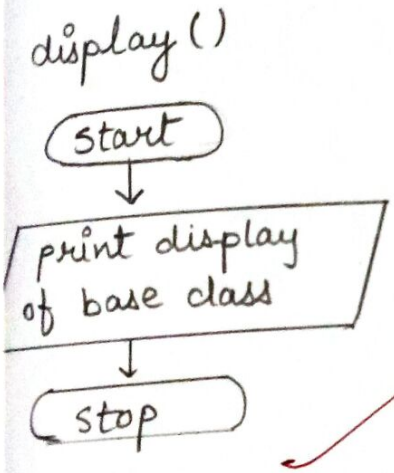
① start
② square s1
③ s1. set_values()
④ s1. cal_sq()
⑤ cube c1
⑥ c1. setvalues()
⑦ c1. cal_cube()
⑧ stop.

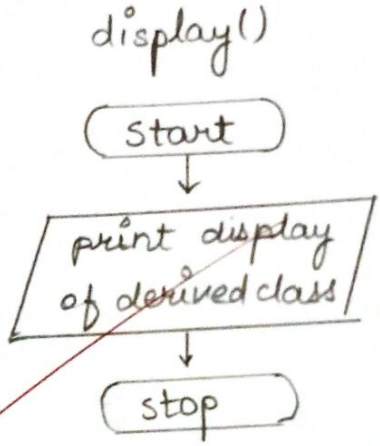
Program 11

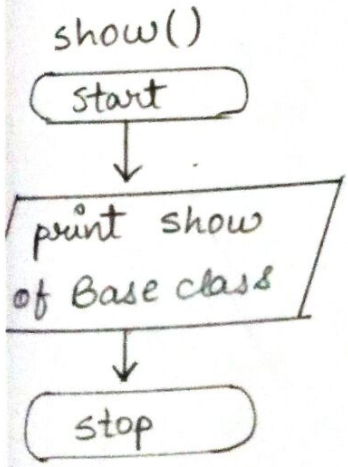
* Flowchart :-

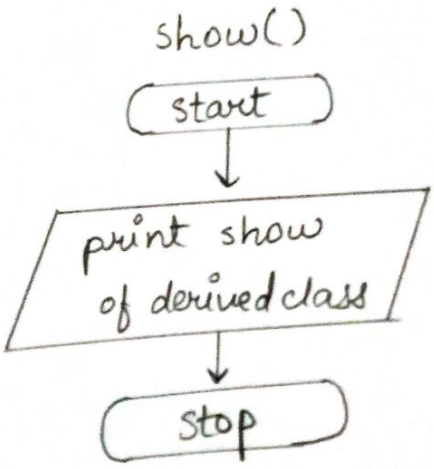












* Algorithms :-

• display()

① start
② print display of base class
③ stop

• show()

① start
② print show of base class
③ stop

• display()

① start
② print display of derived
③ stop.

• show()

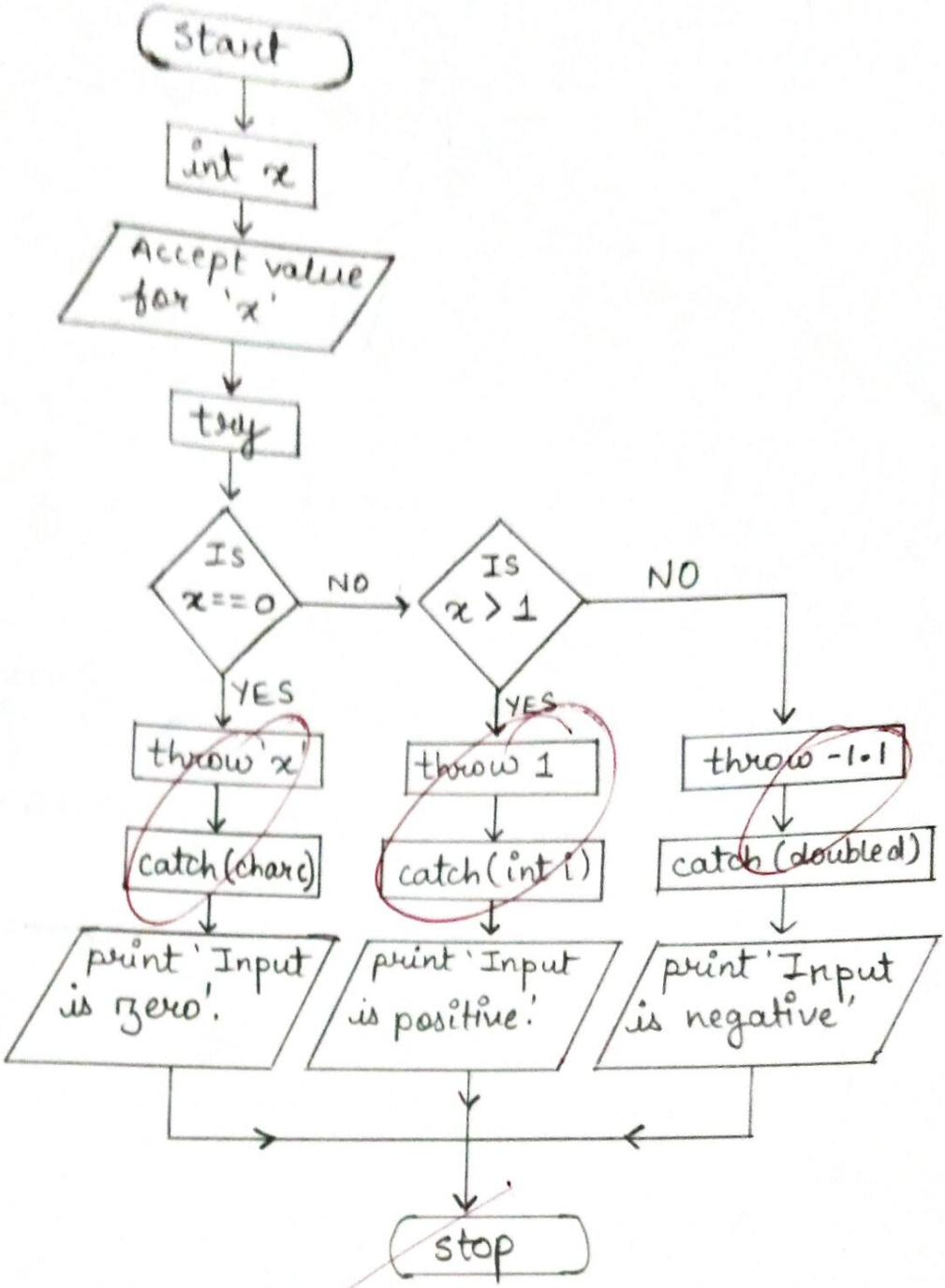
① start
② print show of derived
③ stop.

• main()

① start
② Base OB
Derived OD
③ Base *bptr
④ print bptr points to base
⑤ bptr = &OB
⑥ Call bptr -> display()
⑦ Call bptr -> show()
⑧ print bptr points to derived
⑨ bptr = &OD
⑩ Call bptr -> display()
⑪ Call bptr -> show()
⑫ Stop

// Program 12.

* Flowchart :-

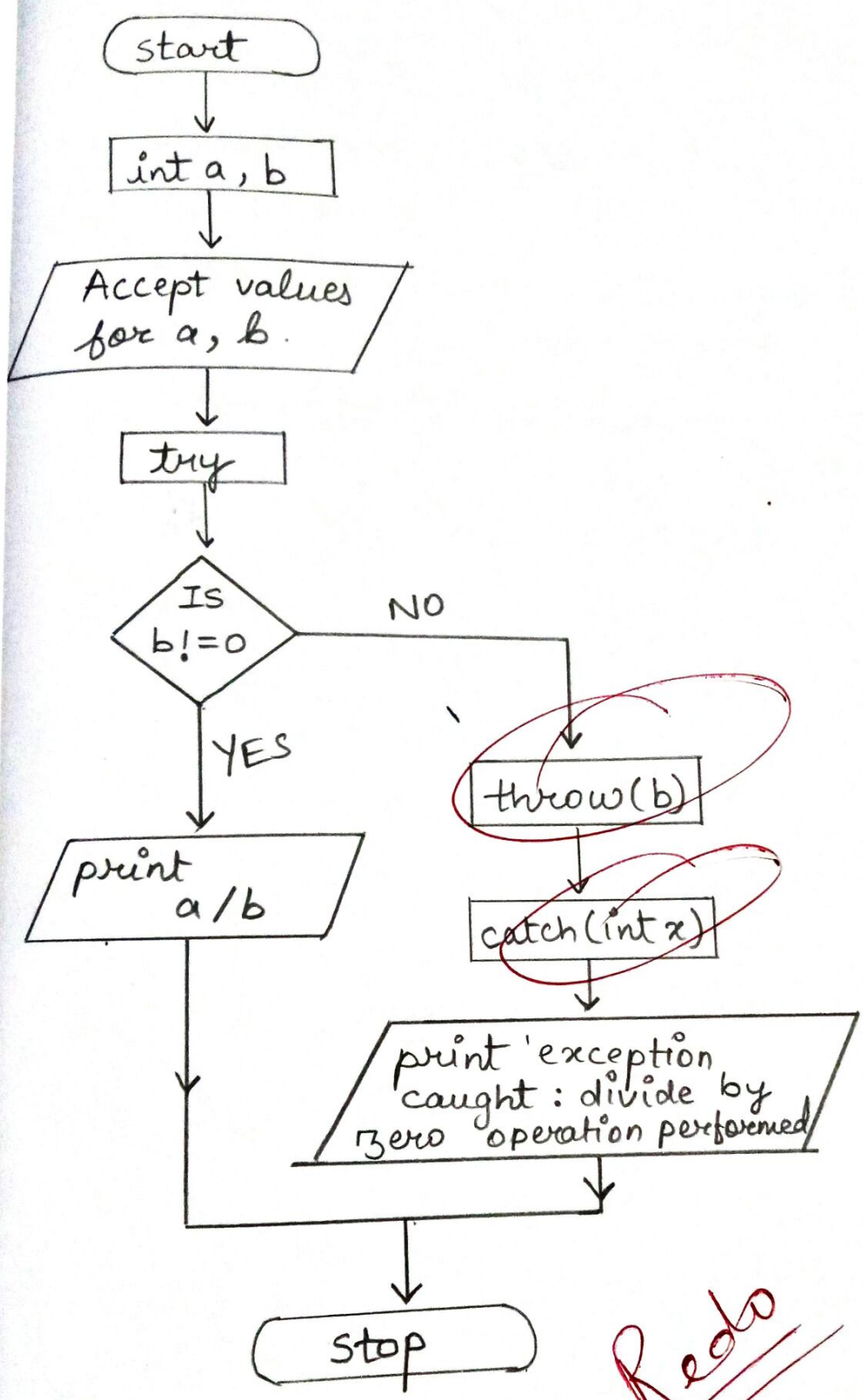


*Algorithm :-

① start
② Declare int x
③ Accept value for x
④ Is x == 0 ?
  YES : (a) Print 'Input is zero'.
        (b) Jump to step 5.
  NO : (a) Is x > 1 ?
       YES : (i) Print 'Input is positive'.
             (ii) Jump to step 5.
       NO : (i) Print 'Input is negative'.
            (ii) Jump to step 5.
⑤ stop

// Program 13

* Flowchart :-



Redo

* Algorithm :-

1. start
2. Declare int a, b.
3. Accept values for a, b.
4. Is b != 0 ?
   - YES : (a) Perform a/b and Display a/b.
     (b) Jump to Step 5.
   - NO : (a) Print " Exception caught : divide by zero operation performed
     (b) Jump to step 5.
5. Stop