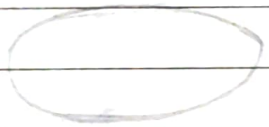
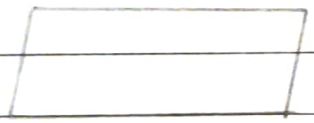
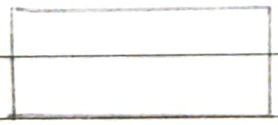
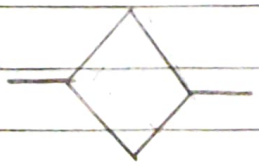
friday

9/9/22

1

Algorithm :- To solve the problem by going through steps in sequence

(1) Start
(2) Input
(3) compute
(4) print
(5) output
(6) End

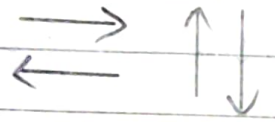
Symbols	Operation	Meaning
1) 	Start / End	Beginning / End of flow chart
2) 	Input / output	Represent the value to be given by the user & the result to be display
3) 	Processing	Represent the arithmetic operation to compute a value
4) 	checking / decision making	logical checking to decide the flow of sequence

5.)

looping — Represent the looping which is represented based on condition value of variable

6)

connector — Represent continuity of the flow chart in another page or place

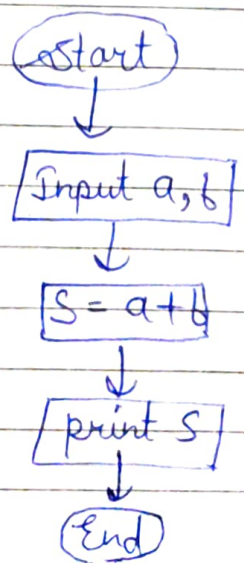
7) 

Arrow — Represent Direction of flow.

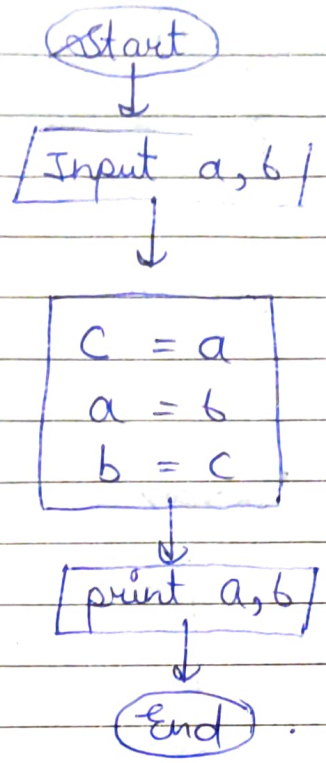
1) <u>Example</u> :- Addition of any two numbers

(1) Start
(2) Read a, b
(3) Compute $s = a + b$
(4) Print S
(5) End

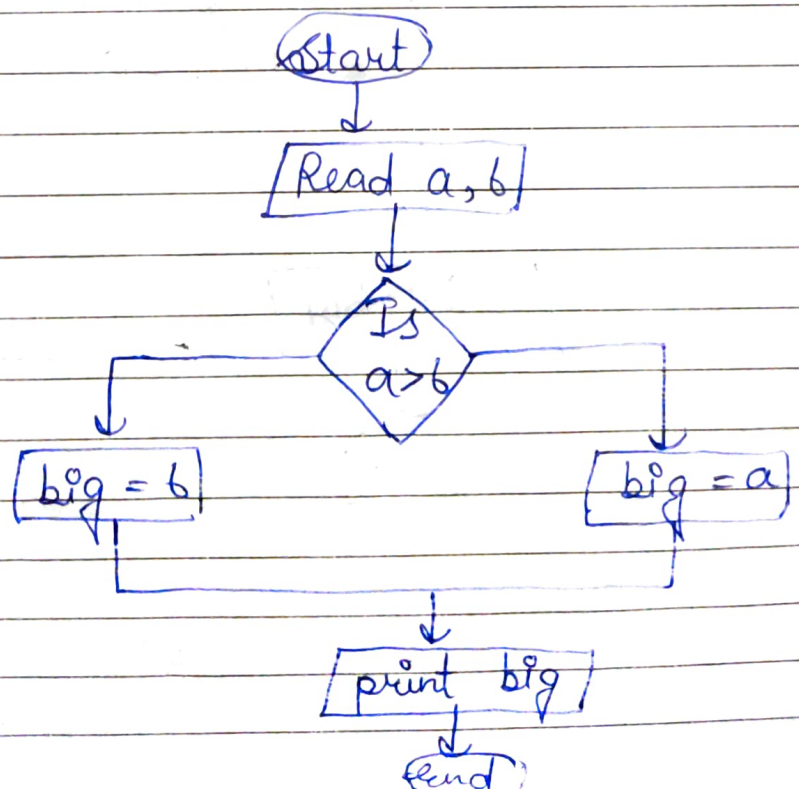
In Algorithm :-



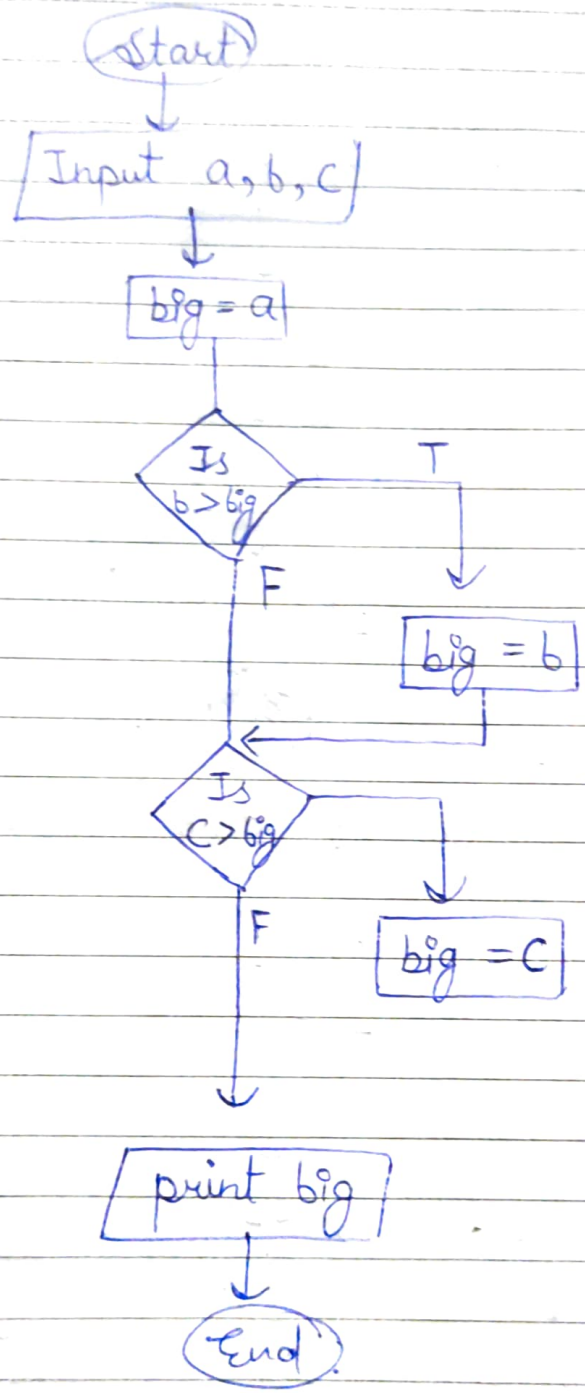
(2) <u>Example</u> :- Swapping of two numbers, a & b.



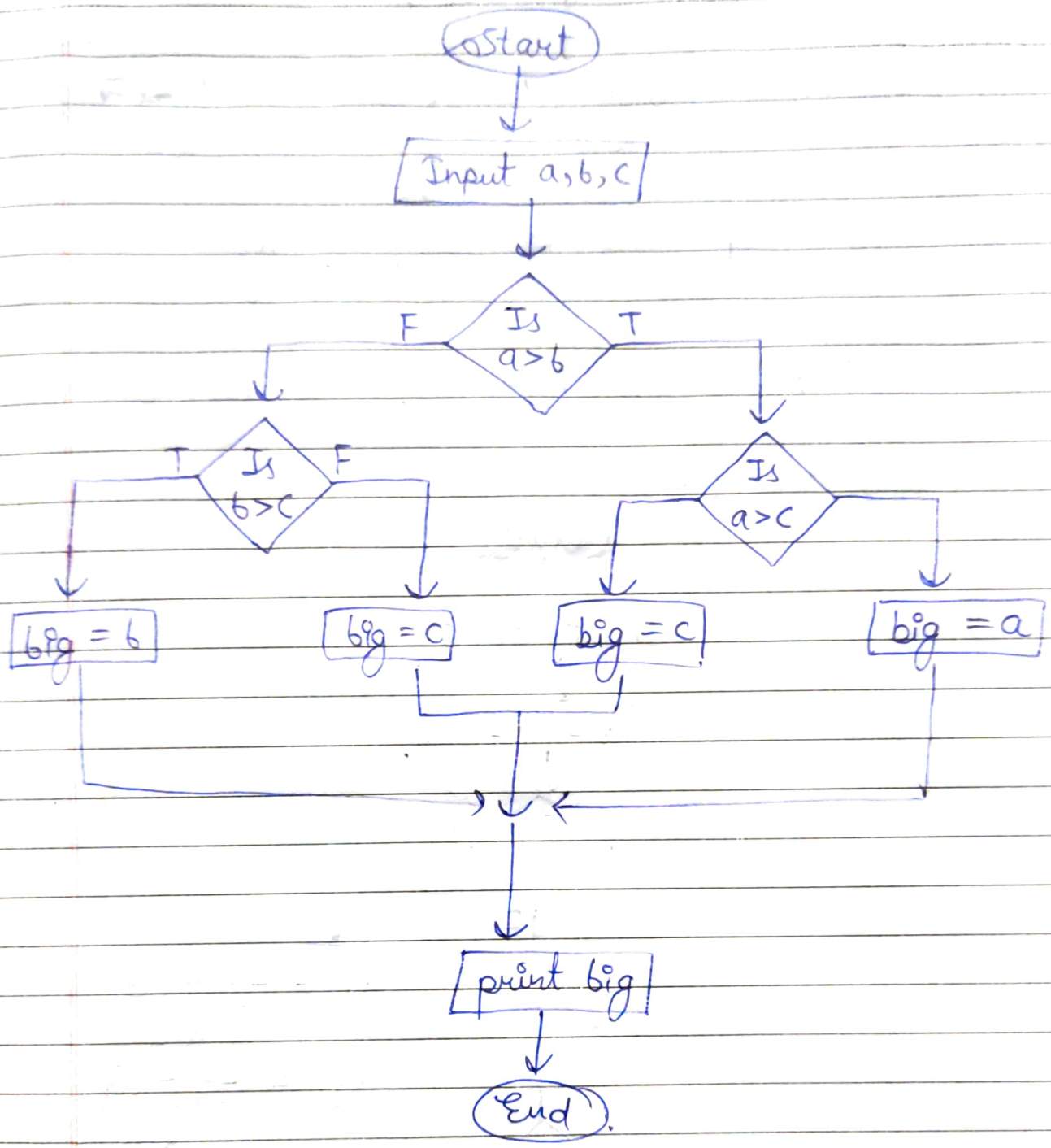
(3) Largest no. among two numbers.



(4) Which is the largest number among 3 numbers a, b, c.



"OR"



(5) Draw a flow chart, to calculate the root of quadratic equation

$a, b, c \rightarrow$ coefficient of quadratic eq^n.

$ax^2 + bx + c \rightarrow$ quadratic eq^n.

$x_1, x_2 \rightarrow$ roots of quadratic eq^n.

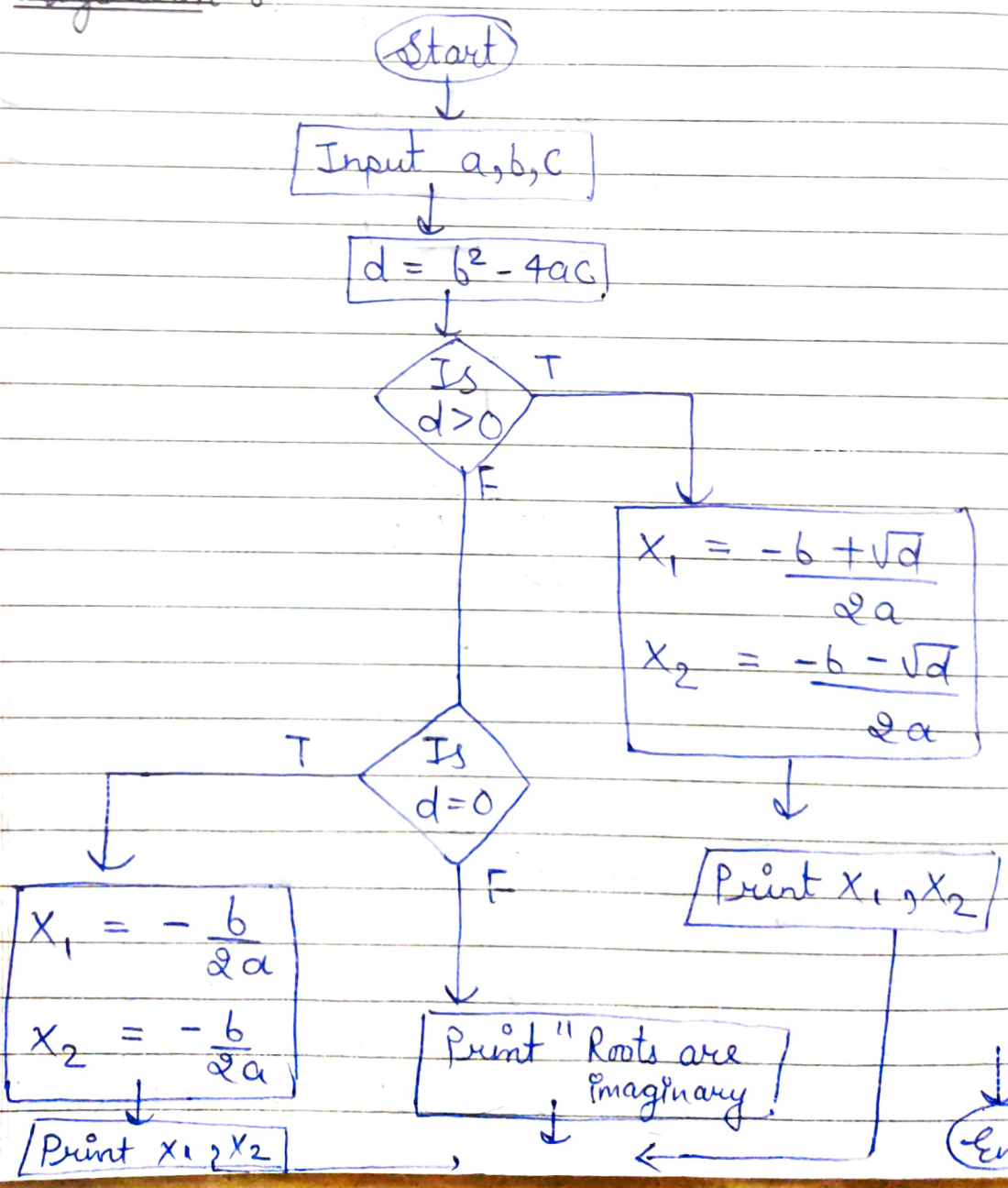
$$\{d = b^2 - 4ac\}$$

$$\left\{X_1 = \frac{-b + \sqrt{d}}{2a}, \quad X_2 = \frac{-b - \sqrt{d}}{2a}\right\} \text{ when } d > 0$$

$$\left\{X_1 = X_2 = \frac{-b}{2a}\right\} \text{ when } d = 0$$

$$\{\text{Roots are imaginary}\} \text{ when } d < 0$$

<u>Algorithm</u> :-



(6) Total salary of Employee ; but total salary depends on basic salary, if basic salary is greater than 30,000/-

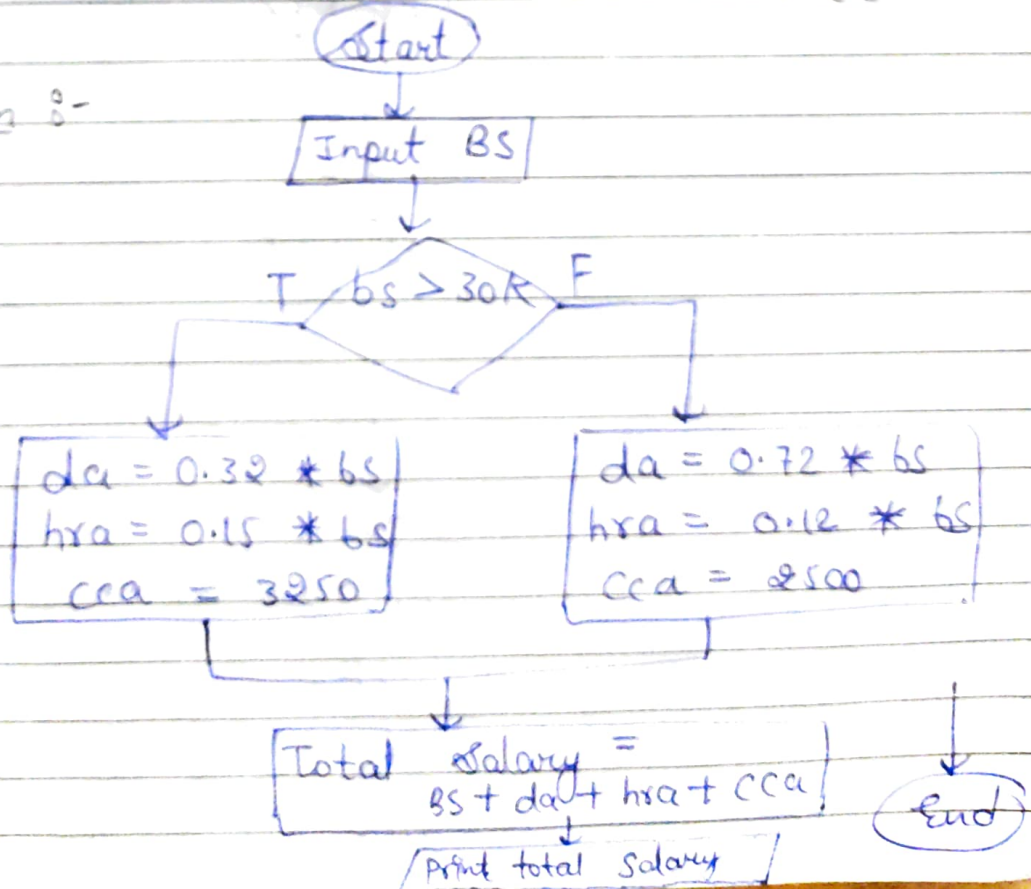
Da = 32% of basic salary
hra = 15% of basic salary
cca = 3250

If basic salary is less than 30,000/-

Da = 72% of basic salary
hra = 12% of basic salary
cca = 2500

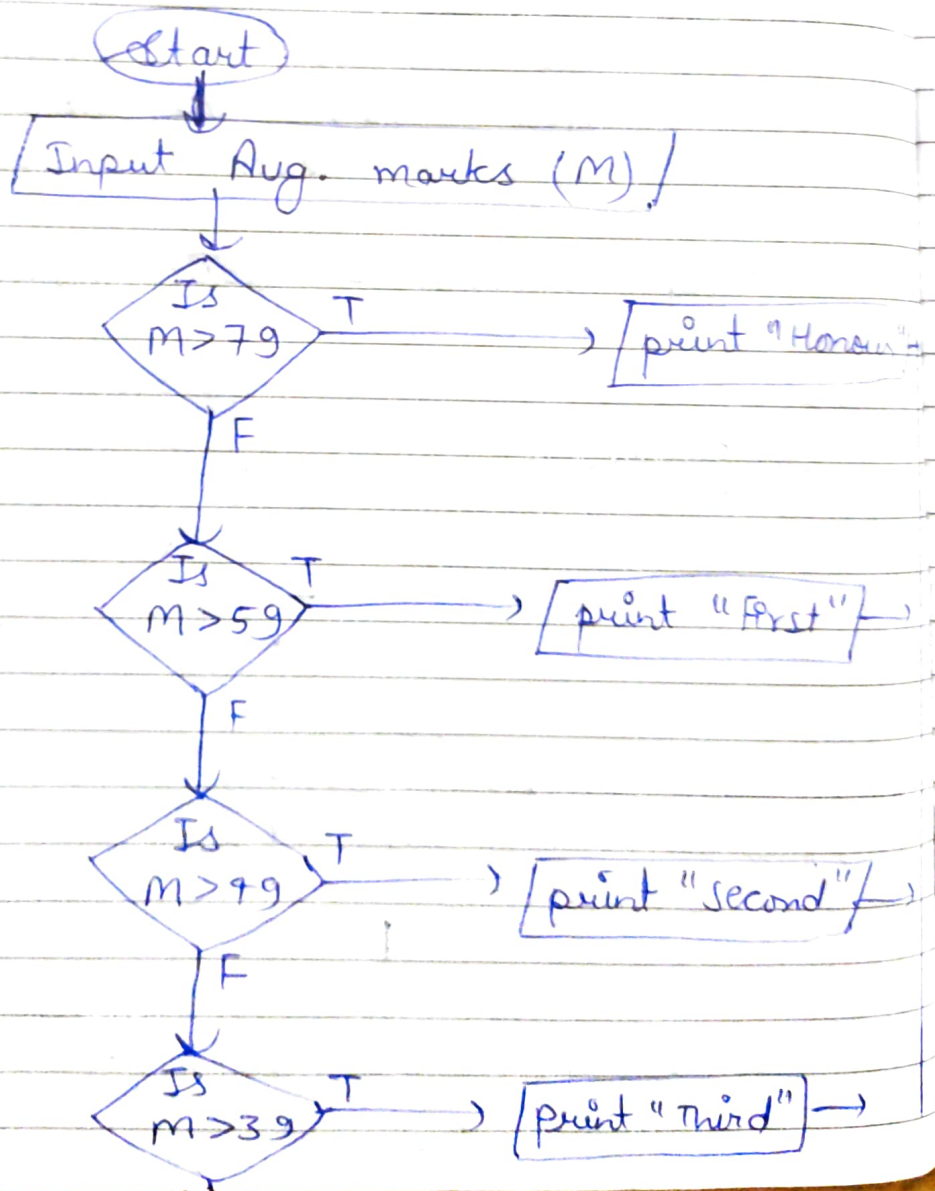
Total salary = Basic salary + Da + hra + cca

<u>Algorithm</u> :-

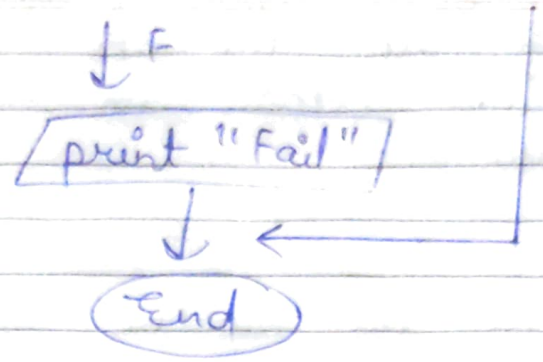


(7) Draw a flow chart to compute the grades According to the average marks.

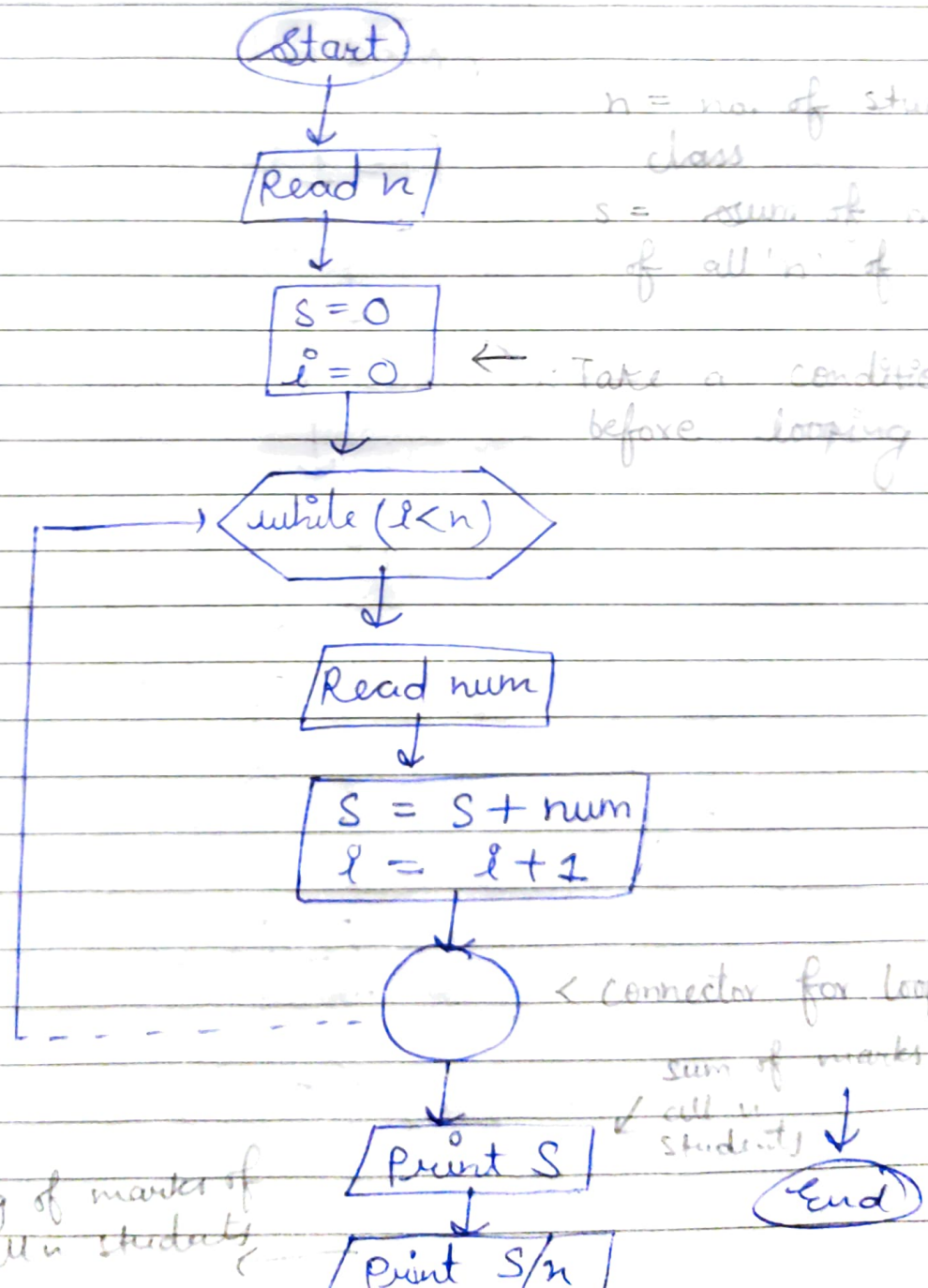
Avg. Marks	Grade
80 - 100	Honor
60 - 79	First
50 - 59	Second
40 - 49	Third
0 - 39	Fail.



the
marks.



Use of looping :-



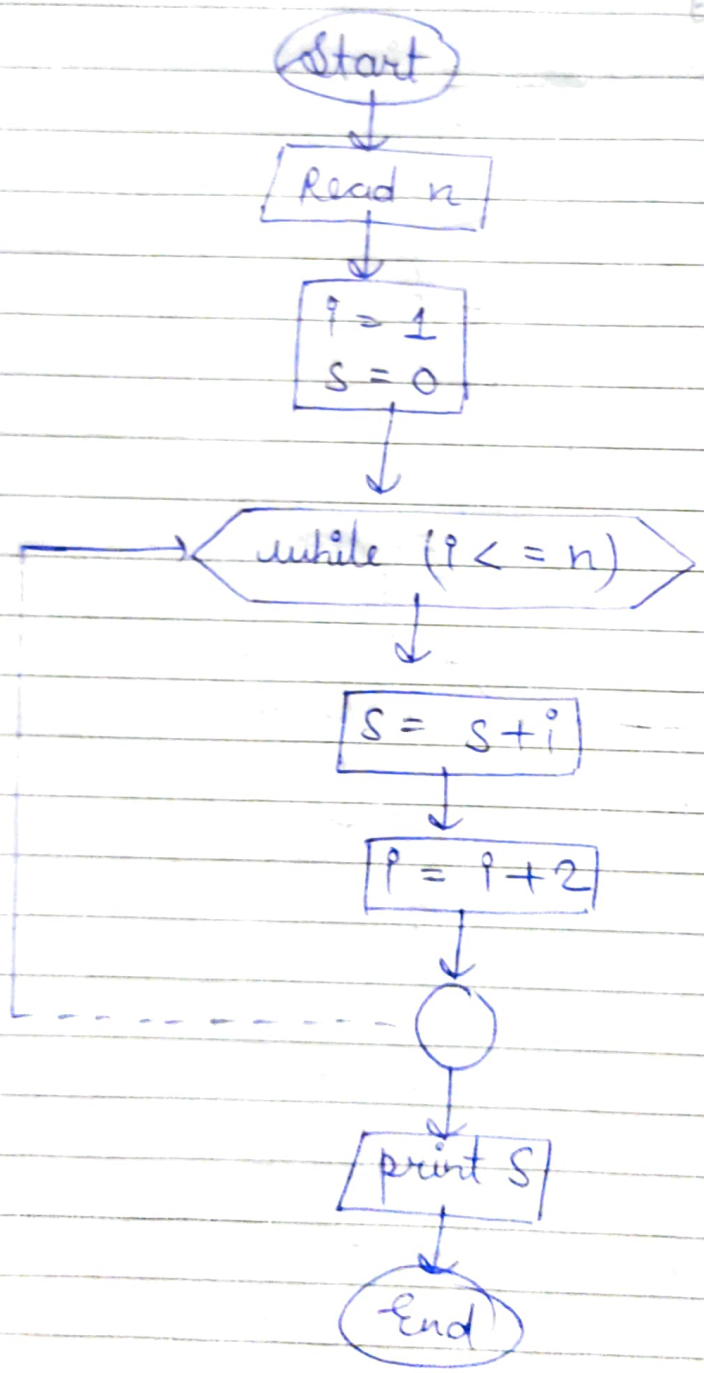
(1) Sum of all the 'n' of students in a particular class. ~~is~~ their Avg. of marks also.

PTO.

(2) Sum of odd number up to 'n'.

Eg:- up to '5'

$n = 5$



$i = 1$
$s = 0$
$1 \leq 5$
$s = 1 + 0 = 1$
$i = 1 + 2 \Rightarrow 3$

$i = 3$
$s = 1$
$3 \leq 5$
$s = 3 + 1 = 4$
$i = 3 + 2 =$
$5 \leq 5$
$(3+1)$
$s = 4 + 5 \Rightarrow$
$i = 5 + 2 =$

character used in C :-

Alphabet = upper case = A to Z
lower case = a to z

↳ Number = 0 to 9

↳ Special characters :-

+	plus	*	asterisk
-	minus	.	full stop
,	comma	;	semi colon
<	less than	>	greater than
_	underscore	~	tilde
:	colon	$	dollar
(	open parenthesis	#	Hash
)	close parenthesis	&	ampersand
/	slash	^	caret
%	percent	|	vertical bar
{	open set bracket	}	close set bracket
!	Exclamation mark	?	Question mark

Identifier :- An identifier is a name having a few characters, numbers and special character (_ underscore). It is used to identify a variable, function, symbolic constant and so on. Identifier can be read in with a maximum of 31 characters. It is a good practice to have identifier with few letters, less than 8 characters.

it is commonly followed with the first letter being an alphabet.

Variable → x2, sum
function → Fact ( ), sum, avg
symbolic constant → PI (π).

Keyword or Reserve word :-

C language uses a following keyword which are not available to users to use them as variable / function name

Keywords are →

(1) auto	(2.) default	(3.) register	(4) float
(5) volatile	(6.) break	(7) do	(8) for
(9) return	(10) switch	(11) while	(12) case
(13) double	(14) goto	(15) short	(16) typedef
(17) char	(18) else	(19) if	(20) signed
(21) union	(22) constant	(23) enum	(24) int
(25) size of	(26) unsigned	(27) continue	(28) extern
(29) long	(30) static	(31) void	(32) struct

Note :- We cannot use these keywords as an variable
↳ fun / symbolic constant.

Constant :- Any unchanged value in a program is called constant.

* There are many type of constant. →

1. Numeric Constant :-

(i) Integer Constant :-
e.g. → -25 , +340 , 42

(ii) Octal Number :- 0 to 7
e.g. → 043 , 0355 , +04.

(iii) Hexa decimal :- [0 - 9] [A - F]
e.g. → 0x 8a , 0x C83
↳ representation of hexadecimal

Real or floating point :-

1. Decimal form :-
e.g. → 0.254 , +342.0 , -3.15

2. Exponential form :-
e.g. → $0.218e^{6}$ ⇒ 0.218×10^{6}
$0.4e^{-32}$ ⇒ 0.4×10^{-32}.

Conversion of hexadecimal and decimal number

1. Convert $(17)_{16}$ into decimal :-

$7 \times 16^{0} + 1 \times 16^{1}$
$7 + 16$ ⇒ 23.

(2) Convert $(23)_{10}$ into hexadecimal :-

16	23		$\Rightarrow (17)_{16}$
	1	7	
	↑ Quotient	↑ Reminder	

String or character :-

character :- A single alphabet or we can say that letter.

eg. → 'y' , 'm'.

String :- " sum = " (It is a collection of letters or characters).

Operators :-

(1) **Airthmetic operator** :- In the C language for the ~~arith~~ arithmetic operators these are :-

Operator	Meaning	Example	Result.
+	Addition	5+2	7
−	Substraction	5-2	3
*	Multiplication	5 * 2	10
/	Division	5/2	2
%	Modulus	5 % 2	1

Example :-

$y = a + b/c - a \% c + 2 * a$

where $a = 5$, $b = 7$, $c = 2$.

$y = 5 + 7/2 - 5 \% 2 + 2 * 5$

$= 5 + 3 - 1 + 10$

$= 8 - 1 + 10$

$= 17$ Ans

(2) Relational operator :- It is used to compare the value of operants.

Relational operators are :-

Operator	Meaning	Example	Result
<	less than	5 < 2	False
>	greater than	5 > 2	True
<=	less than equal to	5 <= 2	False
>=	greater than equal to	5 >= 2	True
==	double equal to	5 == 2	False
!=	Not equal to	5 != 2	True

(3) logical operator :-

operator	meaning	Example	Result
&&	logical AND	(5<2) && (5>3)	False
||	logical OR	(5<2) || (5>3)	True

! logical NOT !(5<e) True

(4) Increment and decrement Operator :-

Increment operator (++) is used to increase the value of a integer and a character variable by 0.

Decreament operator (--) is used to reduce the value of an integer or character variable by 1

Example :- m=15

m++ or ++m will produce the result m=16.

and m=15
m-- or --m will produce the result m=14.

* m++ and m-- are refering the post fix increament and decreament and ++m and --m are refering prefix increament and decreament

* Now here is different meaning of post fix and prefix :-

K = m++, assign a value of m to K

and then increase the value of m by 1.

i.e $k=15$
$m=16$.

Prefix :- $k = ++m$; firstly it will increase the value of m, then after assign that value of m to k

i.e $k=16$
$m=16$.

(5.) **Assignment operator :-**

Operator	Example	Equivalent expression m=15	Result.
+ =	m+ = 10	m = m+10	25
- =	m- = 10	m = m-10	5
* =	m* = 10	m = m*10	150
/ =	m/ = 10	m = m/10	1
% =	m% = 10	m = m%10	5

(6) **Conditional operator :-** (? :) or Termary operator

It is used to check a condition and select a value depending on the value of the condition.

The syntax is :-
Variable = condition ? value 1 : value 2;

when this operator is executed the value of the condition is evaluated if it is true than value 1 will assign to the variable otherwise value 0 can assign to the variable

<u>Example</u> :- big = (a>b) ? a : b ;

If a>b then big = a

If not a>b then big = b.

(7) <u>Bituise Operator</u> :-

Operator	Meaning.
<<	shift the bits to left
>>	shift the bits to right
~	bituise inversion
&	bituise logical AND
|	bituise logical OR
^	bituise exclusive OR

<u>Example</u> :- $a = (13)_{10} = (1101)_2$

$b = (6)_{10} = (110)_2$

a = 00001101

b = 00000110.

(i) a << 2

0	0	0	0	1	1	0	1

$\Rightarrow$ 00110100.

(ii) a >> 2

0	0	0	0	1	1	0	1

$\Rightarrow$ 00000011.

(iii) ~ a

one 1s converted to 0
0 s converted to 1

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

(iv). a & b

00001101
00000110
Ans ↳ 00000100

Condition :-

A	B	A & B
0	0	0
0	1	0
1	0	0
1	1	1

(v) a | b.

00001101
00000110
Ans ↳ 00001111

Condition :-

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

(vi) a ^ b

→ If the bit is same then output 0.
→ If the bit is diff. then output 1.

```
  0 0 0 0 1 1 0 1
  0 0 0 0 0 1 1 0
  ---------------
  0 0 0 0 0 1 0 1 1
```

ans

Condition :-

a	b	and
0	0	0
0	1	1
1	0	1
1	1	0

Incidence order	operator	Associativity
1.	( ) [ ]	L to R
2	++ , -- , -(unary) !~ , * , & size of pointer.	R to L
3.	* / %	L to R
4.	+ -	L to R
5.	<< >>	L to R
6.	< > = >>=	L to R
7.	= = ! =	L to R
8.	&& (bitwise AND)	L to R
9.	^ (bitwise XOR)	L to R
10.	| (bitwise OR)	L to R
11.	&& (logical AND)	L to R

12. || (logical OR) L to R

Example :-

(1) $a == c \mid\mid b > a$

if $a = 3$, $b = 10$, $c = -6$.

$3 == -6 \mid\mid 10 > 3$

F T

T

(2) $b > c$ && $c < 0 \mid\mid a < 0$.

$10 > -6$ && $-6 < 0 \mid\mid 5 < 0$

T T F

T

T.

Variable :- A variable is an identifier or a name which is used to refer a value and this value varies and changes during the program execution. A Variable is written with a combination of letters, numbers and special character (_) underscore.
With the first letter being an alphabet maximum of 31 letters and be used to

write a variable

Example :- X
Fact
C22
total - Sum

Rules of Identifiers :-

1. Upper case and lower case alphabet are taken differently so that variable SUM sum are refering to different values.

2. No special character other than (_) underscore

3. Some C compiler will not accept more than 8 characters.

4. All variable used in C are declared with appropriate datatypes.

5. Reserved word cannot be used as variable

Question :- Which of the following identifiers are invalid ?

total → valid

average_value → Valid
6 Marks → Invalid
cd 200 → Invalid
_ sum → valid
rate $ → Invalid

Basic Datatype :-

Datatype	Byte occupied in RAM	values Ranges	values ranges in decimal
(1) Char	1 byte = 8 bits	-2^7 to $2^7 - 1$	-128 to 127
(2) int	2 bytes = 16 bits	-2^{15} to $2^{15} - 1$	-32768 to 32768
(3) float	4 bytes = 34 bits	-2^{31} to $2^{31} - 1$	$3.4e^{-38}$ to $3.4e^{+38}$
(4) double	8 bytes = 62 bits	-2^{63} to $2^{63} - 1$	$1.7e^{-308}$ to $1.7e^{+308}$

(If we want more than 7 digits we use double)

- Int :- integer.
  float :- floating point value : i.e a number with a fractional part (used for storing single precision floating (4 byte) point number).
  double :- Used for storing double precision floating point value (8 bytes)
  Char :- a single character.

(अगर number को store करना है float में)

(अगर maximum number store करना है double में)

Additional Datatype :-

Datatype	Bytes occupied in RAM	value ranges	value rang in decim
(1) unsigned char	1 byte	0 to 2^8-1	0 to 2
(2) short int	1 byte	-2^7 to 2^7-1	-128 to 12
(3) unsigned short int	1 byte	0 to 2^8-1	0 to 2
(4) unsigned int	2 bytes	0 to $2^{16}-1$	0 to 65
(5) long int	4 bytes	-2^{31} to $2^{31}-1$	-2147483648
(6) unsigned long int	4 bytes	0 to $2^{32}-1$	0 to 4
(7) long double	10 bytes	-2^{79} to $2^{79}-1$	- 604 to

~~Unsigned int :- positive integer~~

* Additional Operator :-

(1) size of operator :- (It returns the size of data.) It is used to find the no. of bytes occupied by a variable datatype in the Computer memory.

Example :- int m , x [50].

size of (float) ; = 4 bytes
size of (int) ; = 2 bytes
size of (m) ;
size of (x) ;
size of (marks) ;

size of (avg) ;

(2) <u>Comma operator</u> :- It is used to link related expression to make the program more compact related expression are evaluated from L to R and the value of a right most expression is written as the value of comma expression

2147483647
97295
909,807,314,587,353,088
462,909,807,314,587,353,087

<u>Example</u> :-

```
temp = x ;
x = y ;
y = temp ;
```

↳ temp = x , x = y , y = temp ;

<u>Mixed mode operation</u> :-

low Char → int → long int → float → double high

<u>Example</u> :- int m = 15 ;
float x = 3.1 , y ;

y = m * x ;
float int float.