FIRST_SEM FRESHMAN YEAR

C Programming

Report II

Bishal Khatri
ID No.: 201611
SOFTWARE ENGINEERING
MORNING SHIFT

C Programming Operators

An operator is a symbol that operates on a value or a variable. For example: `+` is an operator to perform addition.
C has a wide range of operators to perform various operations.

C Arithmetic Operators

An arithmetic operator performs mathematical operations such as addition, subtraction, multiplication, division etc on numerical values (constants and variables).

Operator	Meaning of Operator
+	addition or unary plus
-	subtraction or unary minus
*	multiplication

Operator	Meaning of Operator
/	division
%	remainder after division (modulo division)

Example 1: Arithmetic Operators

```
// Working of arithmetic operators
#include <stdio.h>
int main()
{
    int a = 9,b = 4, c;

    c = a+b;
    printf("a+b = %d \n",c);
    c = a-b;
    printf("a-b = %d \n",c);
    c = a*b;
    printf("a*b = %d \n",c);
    c = a/b;
    printf("a/b = %d \n",c);
    c = a%b;
    printf("Remainder when a divided by b = %d \n",c);
```

Output

```
a+b = 13
a-b = 5
a*b = 36
a/b = 2
Remainder when a divided by b=1
```

The operators `+`, `-` and `*` computes addition, subtraction, and multiplication respectively as you might have expected.

In normal calculation, `9/4 = 2.25`. However, the output is `2` in the program.
It is because both the variables `a` and b are integers. Hence, the output is also an integer. The compiler neglects the term after the decimal point and shows answer `2` instead of `2.25`.
The modulo operator `%` computes the remainder. When `a=9` is divided by `b=4`, the remainder is `1`. The `%` operator can only be used with integers.
Suppose `a = 5.0`, `b = 2.0`, `c = 5` and `d = 2`. Then in C programming,

```
// Either one of the operands is a floating-point number

a/b = 2.5

a/d = 2.5

c/b = 2.5



// Both operands are integers

c/d = 2
```

C Increment and Decrement Operators

C programming has two operators increment `++` and decrement `--` to change the value of an operand (constant or variable) by 1.
Increment `++` increases the value by 1 whereas decrement `--` decreases the value by 1. These two operators are unary operators, meaning they only operate on a single operand.

Here, the operators `++` and `--` are used as prefixes. These two operators can also be used as postfixes like `a++` and `a--`.

C Assignment Operators

An assignment operator is used for assigning a value to a variable. The most common assignment operator is `=`

Operator	Example	Same as
=	a = b	a = b
+=	a += b	a = a+b
-=	a -= b	a = a-b
*=	a *= b	a = a*b
/=	a /= b	a = a/b
%=	a %= b	a = a%b

C Relational Operators

A relational operator checks the relationship between two operands. If the relation is true, it returns 1; if the relation is false, it returns value 0.

Relational operators are used in decision making and loops.

Operator	Meaning of Operator	Example
==	Equal to	`5 == 3` is evaluated to 0
>	Greater than	`5 > 3` is evaluated to 1
<	Less than	`5 < 3` is evaluated to 0
!=	Not equal to	`5 != 3` is evaluated to 1
>=	Greater than or equal to	`5 >= 3` is evaluated to 1
<=	Less than or equal to	`5 <= 3` is evaluated to 0

Explanation of logical operator program

- `(a == b) && (c > 5)` evaluates to 1 because both operands `(a == b)` and `(c > b)` is 1 (true).
- `(a == b) && (c < b)` evaluates to 0 because operand `(c < b)` is 0 (false).
- `(a == b) || (c < b)` evaluates to 1 because `(a = b)` is 1 (true).
- `(a != b) || (c < b)` evaluates to 0 because both operand `(a != b)` and `(c < b)` are 0 (false).
- `!(a != b)` evaluates to 1 because operand `(a != b)` is 0 (false). Hence, !(a != b) is 1 (true).

- `!(a == b)` evaluates to 0 because `(a == b)` is 1 (true). Hence, `!(a == b)` is 0 (false).

C Bitwise Operators

During computation, mathematical operations like: addition, subtraction, multiplication, division, etc are converted to bit-level which makes processing faster and saves power.

Bitwise operators are used in C programming to perform bit-level operations.

Operators	Meaning of operators
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
~	Bitwise complement
<<	Shift left
>>	Shift right

Other Operators

Comma Operator

Comma operators are used to link related expressions together. For example:

```
int a, c = 5, d;
```

The sizeof operator

The `sizeof` is a unary operator that returns the size of data (constants, variables, array, structure, etc).

```
/*
Q.1 Algorithm
Step1: Start
Step2: Declare A, B variables.
Step3: Calculate A=++A + ++A, B=--A - --B
Step4: Display A and B.
Step5: Stop
*/
/*
1.What will be the value of A and B on each line?
A=10, B= 5; A=++A + ++A; B=A-- - --B; Also explain the operations.
*/
/*
```

```
#include<stdio.h>
int main()
{
int A = 10, B = 5;
A=++A + ++A;
B=A-- - --B;
printf("A is %d ", A);
printf("and B is %d ", B);
return 0;
}
*/
//Output: A is 23 and B is 20


/*
Q.2 Algorithm to find out whether the entered number is odd or even using conditional operator.*/
/*
Step1: Start
Step2: Declare a variable name as B.
Step3: Read input variable B and assign input value.
Step4: Check whether divisible by 2 or not.
Step4a: If divisible print even
Step4.b: If not divisible print odd
Step5: Stop
*/
```

```
/*
Q.2-Write a program to find out whether the entered number is odd or even using conditional operator?
*/
/*
#include<stdio.h>
int main(){
int num;
printf("Enter the value of num: ");
scanf("%d", &num);
(num%2==0)?
printf("Given number is even"):
printf("Given number is odd");
return 0;
}
*/
//When num is 6, Given number is even.
//When num is 3, Given number is odd.

/*
Q.3 Algorithm to find maximum of three numbers using conditional operator.
Step1: Start
Step2: Declare the variables name as a, b and c.
Step3: Read a, b and c.
Step4: Check if a>b if 1(true), then go through  a>c  if 1(true) print a is max.
Step5: If 0(false)  c is max.
Step6: if c is min go through b>c, if 1(true), print b is max.
Step7: If 0(false) print c is max.
Step5: stop
*/


/*
Q.3-Write a program to find maximum of three numbers using conditional operator?*/
/*
#include<stdio.h>
int main(){
int a, b, c, max_num;
printf("Enter the three numbers: ");
scanf("%d%d%d",&a,&b,&c);
max_num=(a>b&&a>c)?
a:(b>a&&b>c)?b:c;
printf("Max number is %d ",max_num);
return 0;
}
*/
```

//when 7 5 0, Max number is 7.

/*
Q.4 Algorithm to find out the real and equal roots of a quadratic equation using conditional operator.

Step1: Start
Step2: Input coefficients of quadratic equation from user. Store it in some variable say a, b and c.
Step3: Find discriminant of the given equation, using formula discriminant = (b*b) - (4*a*c).
Step4: Compute roots based on the nature of discriminant.
Step5: If discriminant > 0 then, root1 = (-b + sqrt(discriminant)) / (2*a) and root2 = (-b - sqrt(discriminant)) / (2*a).
Step6: If discriminant == 0 then, root1 = root2 = -b / (2*a).
Step7: Else if discriminant < 0 then, there are two distinct complex roots where root1 = -b / (2*a) and root2 = -b / (2*a).
Step8: Stop
*/
/*
Q.4 source_code
#include <stdio.h>
#include <math.h>
int main() {
float a, b, c; float root1, root2, imaginary; float discriminant;
printf("Enter values of a, b, c of quadratic equation (aX^2 + bX + c): "); scanf("%f%f%f", &a, &b, &c);
discriminant = (b * b) - (4 * a * c);
if(discriminant > 0)
{
root1 = (-b + sqrt(discriminant)) / (2*a);
root2 = (-b - sqrt(discriminant)) / (2*a); printf("Two distinct and real roots exists: %.2f and %.2f", root1, root2);
}
else if(discriminant == 0)
{
root1 = root2 = -b / (2 * a);
printf("Two equal and real roots exists: %.2f and %.2f", root1, root2);
}
else if(discriminant < 0)
{ root1 = root2 = -b / (2 * a);
imaginary = sqrt(-discriminant) / (2 * a);
printf("Two distinct complex roots exists: %.2f + i%.2f and %.2f - i%.2f", root1, imaginary, root2, imaginary);
}
return 0;
}
*/
/*Enter values of a, b, c of quadratic equation (aX^2 + bX + c): 4 7 1

Two distinct and real roots exists: -0.16 and -1.59
*/

/*
Q.5-Write a program to illustrate the modulus operator in which second is given as an input and the program converts it to hours, minutes and seconds?
Q.5 Algorithm

Step1: Start
Step2: Read a variable name as x in sec unit.
Step3: Print variable x
Step4: Compute ue1= x/3600, ue2= (x%3600)/60 and ue3= (x%3600)%60
Meaning ue1, ue2, ue3 represents time in hours, minutes and second.
Step5: Display output.
Step6: Stop
*/
//Q.5 source_code
/*
#include<stdio.h>
int main()
{
int ue1,ue2,ue3,enput;
printf("Time in second: ");
scanf("%d",&enput);
ue1=enput/3600;
ue2=(enput%3600)/60;
ue3=(enput%3600)%60;
printf("The time is %d hours %d minutes and %d second.",ue1,ue2,ue3);
return 0;
}
//Time in second: 89007
//The time is 24 hours 43 minutes and 27 seconds
*/

/*
6. Write a program that asks for your height in feet and inches and your weight in kilograms (use three variables to store the information). Convert your height in feet and inches to your height in inches. Then convert your height in inches to height in meter by multiplying it by 0.0254. Now divide your weight by square of your height in meter and finally assign the output to variable ratio. Also display all the information.*/

/*

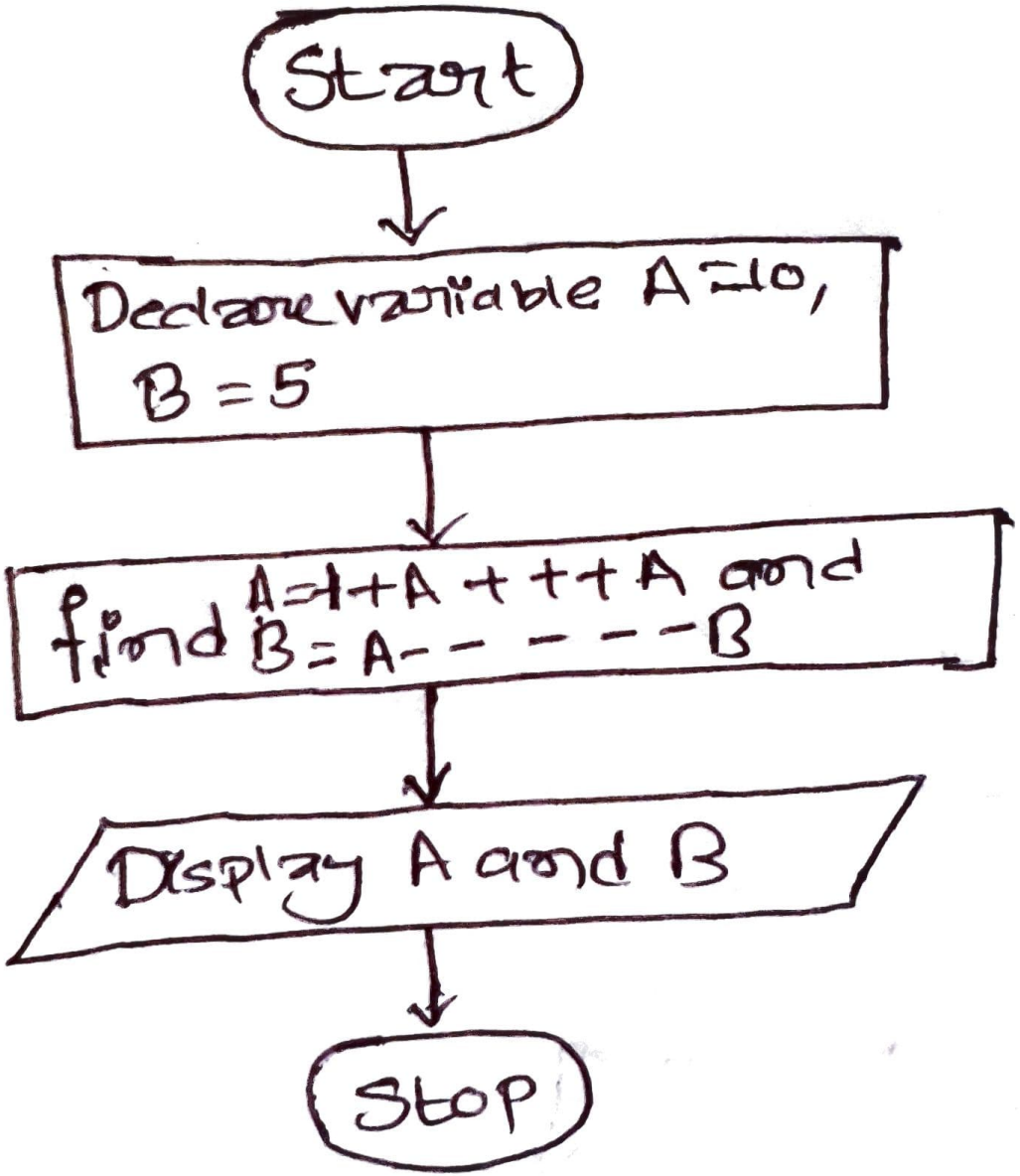
Algorithm for question no.6.
Step1: Start
Step2: Declare variables name as x, y and z with respective units as mentioned in question.
Step3: Read input variables x, y and z.
Step4: Compute a = x*12+y, b = a*0.0254, c = z/b².
Step5: Print the values of a, b and c.
Step6: Stop

//source_code

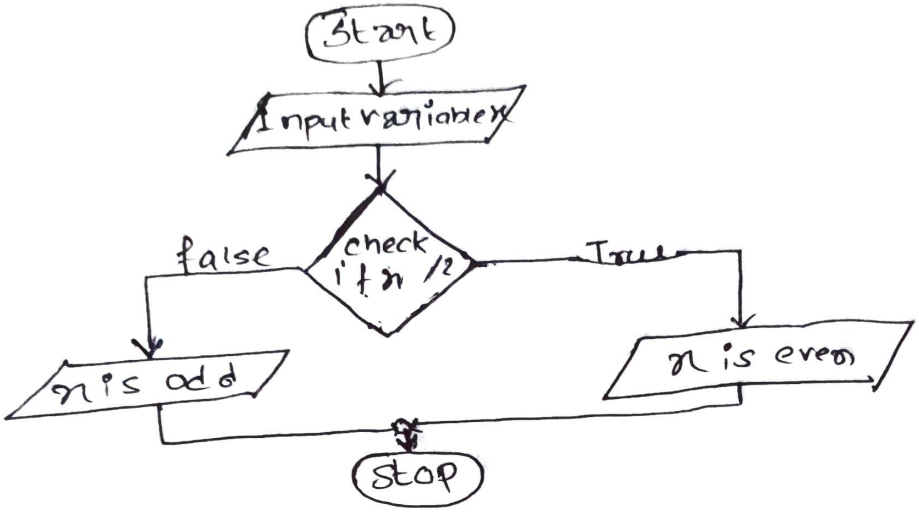
```
/*
#include<stdio.h>
int main()
{
int x, y, z, a;
double b, c;
printf("Enter the height in feet and inches :");
scanf("%d %d", &x, &y);
printf("Enter the weight in kilograms: ");
scanf("%d", &z);
a= x*12+y;
printf("Height in inches is %d inches \n ", a);
b= a*0.0254;
printf("Height in meter is %.3lf meters \n", b);
c = z/(b*b);
printf("Ratio is %.3lf ", c);
return 0;
}
*/
```

/* When height in feet and inches: 18 17 and weight in kilograms is 70 then it shows Height in inches is 233 inches and in meter is 5.918 meter and ratio is 1.999*/

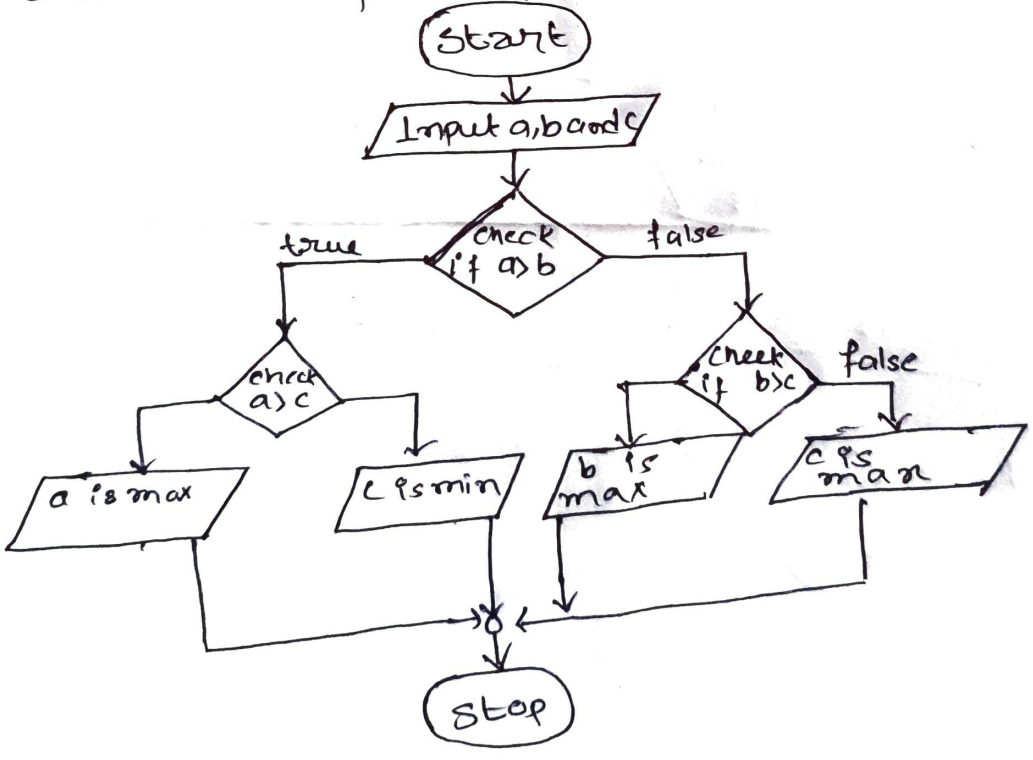
Q.1 flowchart



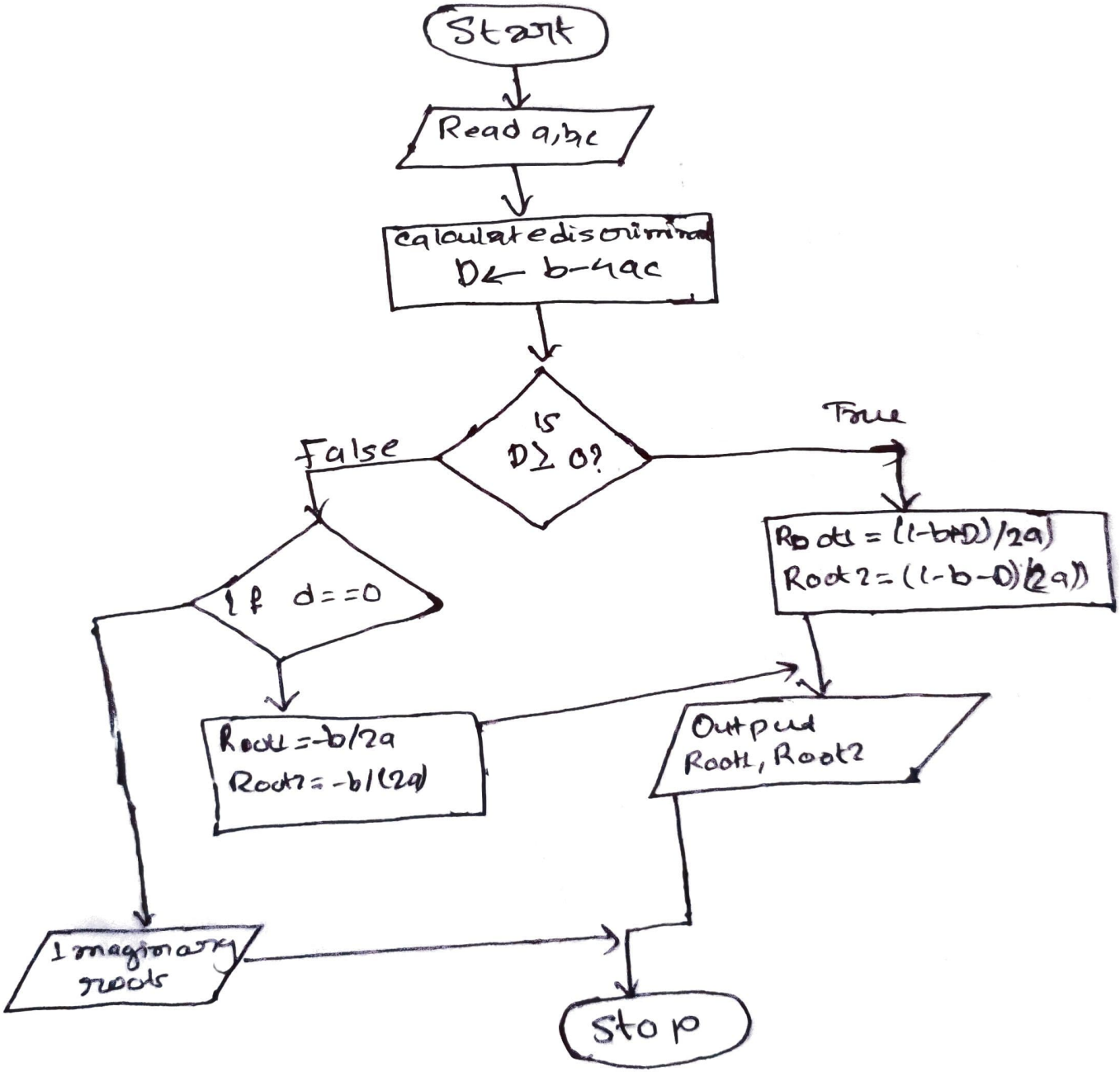
Q.2. flowchart to find out whether the entered number is odd or even using conditional operator.



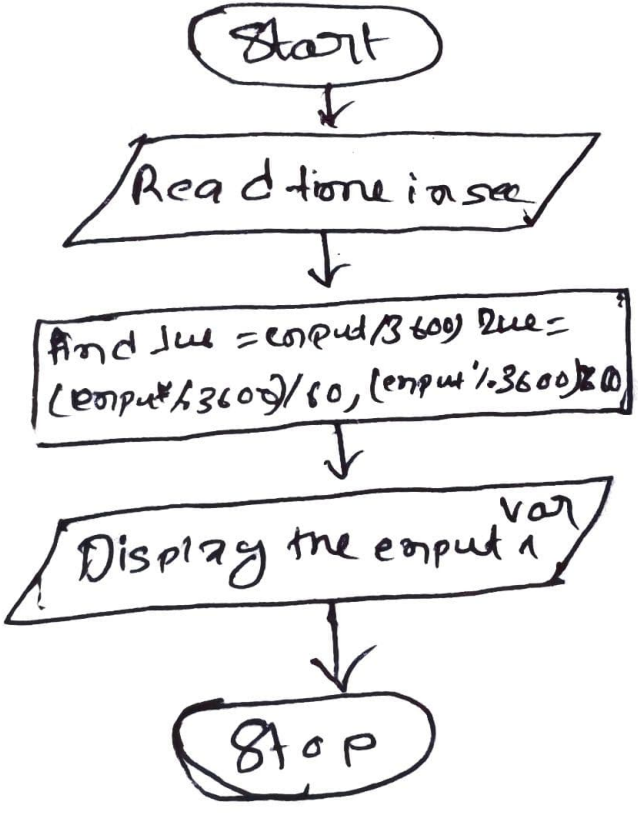
Q.3 flowchart to find maximum of three numbers using conditional operator.



Q. 4 flow chart to find the roots of a quadratic eqn using conditional operator.



Q 5. flowchart to illustrate the modulus operator in which second is given as an input and the program converts it to hours, minutes and seconds.



Q. 6.

flowchart