Module 2 POP C- Programming Solved ALL Model Question Paper

Compter Science and Engineering (CMR Institute of Technology)

2022 scheme
2023 VTU exam

Module-02 of POP C-language BPOP5103

solved 3 Model-Question paper

ARUN'S
PAGE NO.
DATE / /

theory:-

1. Explain with syntax, if and if-else statements in C program.

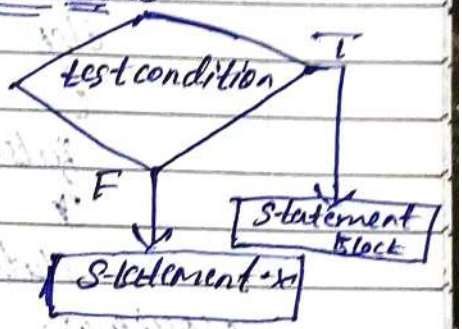
Ans. (1) if :-

Definition:- The if-statement is a simple selection/decision statement when a set of statements have to be executed or skipped. When the condition is true or false, then if-statement is used.

Syntax:-

```
if (test condition)
{
  statement-block;
}
statement-x;
```

Flowchart



Working:-

The condition is evaluated, if it is true the statement-block is executed, otherwise the statement-block will be skipped & the execution will jump to the statement-x.

example:-

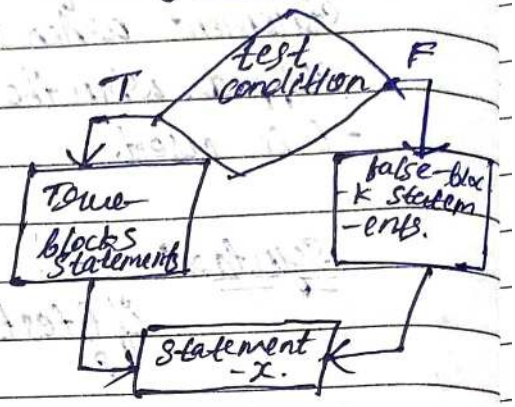
```
#include <stdio.h>
{ int n=10;
  if (n%2==0)
  {
    printf("even number");
  }
}
```

② If-else:-

definition:-

It is a two-way decision statement which selects the statements from two distinct alternatives.

flowchart



syntax:-

```
if (test condition)
{
true-block statement(s);
}
else
{
false-block statement(s);
}
statement-x;
```

working:-

If the test condition is true, then the true-block statement(s), immediatly following the if statements are executed, otherwise the false-block statements are executed.

ex:-

```
#include<stdio.h>
void main(){
int n=10;
if(n%2==0)
{
printf("even number");
}
else
{
printf("odd number");
}
}.
```

* in additional.

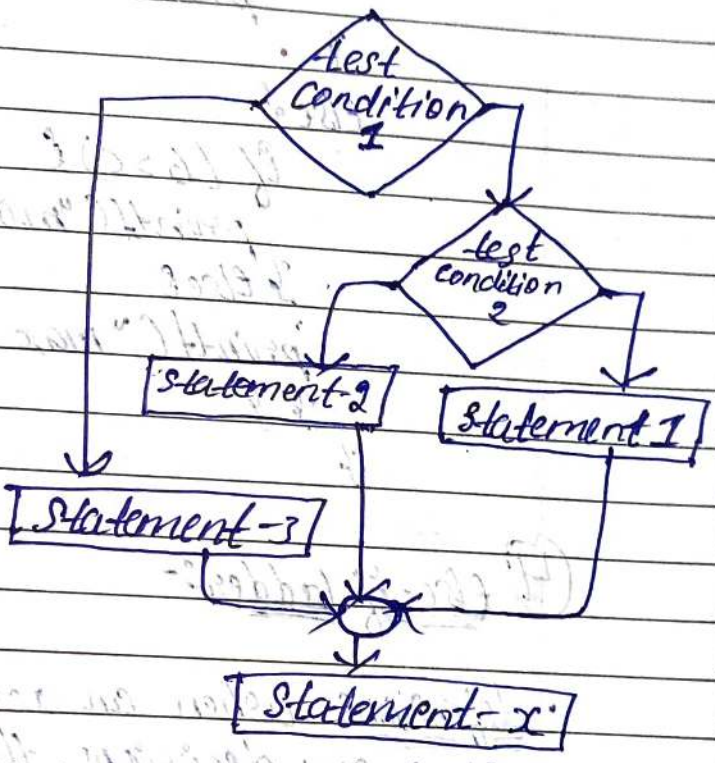
③ nested if-else:-

- definition:- An if or if else statement within another if or if-else statement is called nested if statement.
  * when an action has to be performed based on many decisions then this statement is used.
  so, it is called multi-way decision statement.

- Syntax:-

```
if (test condition 1)
{
if (test condition 2)
{
  statement-1;
}
else
{
  statement-2;
}
}
else
{
statement-3;
}
statement-x;
```

Flowchart:-



- working:-
  if the condition-1 is false, the statement-3 will be executed, otherwise it continues to perform the second test, if the condition-2 is true, the statement-1 will be executed, otherwise the statement-2 will be executed & then control is transferred to student-x

ARUN'S
PAGE NO.
DATE / /

ex:- program to find longest of 3 numbers:-

```
#include<stdio.h>
void main()
{
float a,b,c,max;
printf("enter the value of a,b,and c\n");
scanf("%f%f%f",&a,&b,&c);
if(a>b)
 { if(a>c){
 printf("max=%f\n",a);
}else{
      printf("max=%f\n",c);
 }
 else{
    if(b>c){
     printf("max=%f\n",b);
    }else{
    printf("max=%f\n",c);
    }}
}
```

(4) else-if ladder:-

- definitions- when an action has to be performed based on many decisions, then this statement is used. so it is called multi-way decision statement.

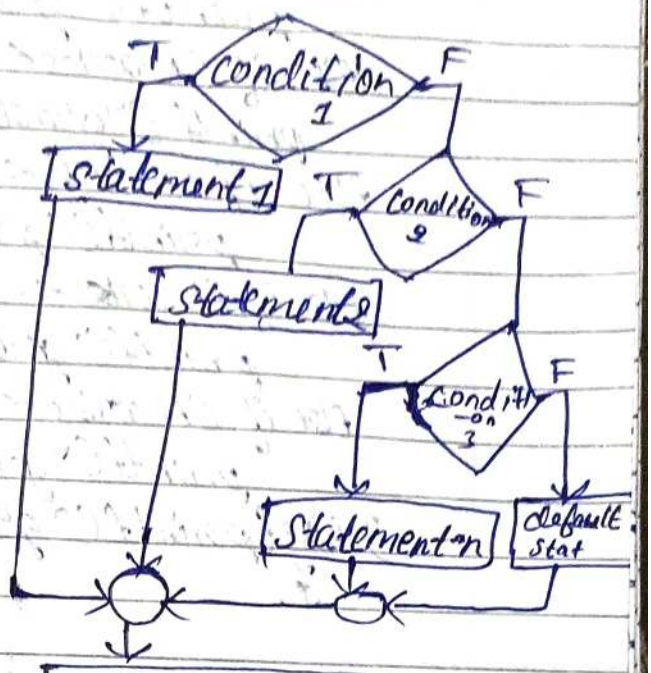
  * The else-if ladder is a form of nested if-statement. here nesting is allowed only in the else-part. the orderly nesting of if-statement only in the else part is called else-if ladder.

- Syntax:-

```
if(condition 1){
   statement-1;
else-if(condition 2)
   statement-2;
```

```
else if (condition n)
    statement-n;
else
    default-statement;
}
statement-x;
```

flow chart



* working:-

The conditions are evaluated from the top, downwards as soon as a true condition is found, the statement associated with it is executed and the control is transferred to the statement-x (skipping the rest of the ladders).

* when all the n conditions become false, then the final else containing the default statement will be executed.

ex:- program to display the grade obtained by a student based on the marks.

the relation b/w the grades and marks is shown

0-34	F (fail)	60-69	C	90-100	O (outstanding)
35-44	E	70-79	B		
45-59	D	80-89	A		

```
#include<stdio.h>
void main()
{
    int marks;
    printf("enter the marks \n");
    scanf("%d", &marks);
    if (marks<=34){
    printf("grade F \n");
```

```
else if (marks<=49)
    printf("grade E\n");
else if (marks<=59)
    printf("grade D\n");
else if (marks<=69)
    printf("grade C\n");
else if (marks<=79)
    printf("grade B\n");
else if (marks<=89)
    printf("grade A\n");
else printf("grade O - outstanding");
}
```

2. Explain switch statement with syntax and example.

Ans:- definition:-

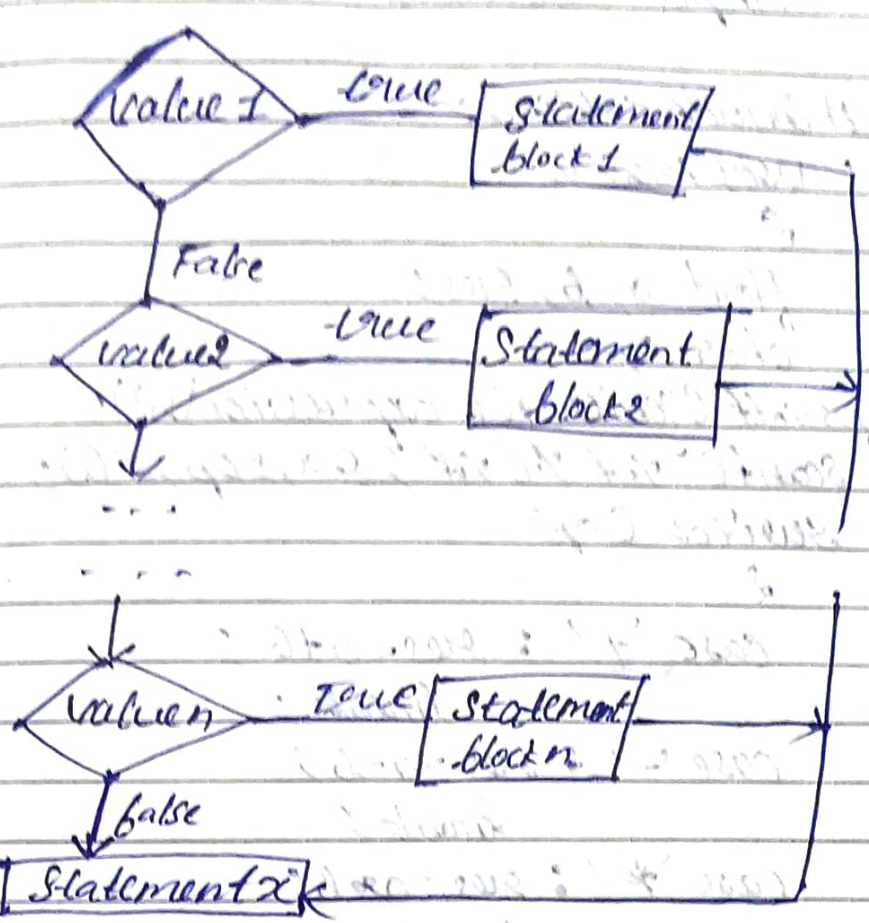
* The 'switch' statement is a control statement used to make one alternative among several alternatives.

* It is a "multi-way decision statement" that whether an expression matches one of the case values and branches accordingly.

- Syntax:-

```
switch (expression)
{
case value-1 : statent block 1;
        break;
...
case value-n : statement blak n;
        break;
default : statements;
}
statement x;
```

Flow chart:-



Working:-

* first the expression within switch is evaluated
* the value of an expression is compared with all the case values.
* the value of an expression within switch is compared with the case value-1. if it matches then the statements associated with that case are executed. if not then case value-2 is compared, if it matches then associated statements are executed and so-on.
* The "default" statements are executed when no match is found.
* A default is optional.

* **break:-** the 'break' statement causes an exit from the switch. it indicates end of a particular case and causes the control to come out of the switch.

example:- program to simulate the functions of a simple calculator

```
#include <stdio.h>
void main()
{
float a, b, res;
char op;
printf("enter the expression\n");
scanf("%f %c %f", &a, &op, &b);
switch (op)
{
case '+' : res=a+b;
           break;
case '-' : res=a-b;
           break;
case '*' : res=a*b;
           break;
case '/' : if (b!=0){
           res=a/b; }
           else{printf("div by zero\n");
           exit(0);
           }
           break;
default : printf("illegal operator\n");
          exit(0);
}
printf("%f %c %f = %f", a, op, b, res);
}
```

3. (b) Distinguish between the break and continue statement.

Ans → Next page.

break statement	Continue statement
1. when break is executed the statements following break are skipped & cause the loop to be terminated.	1. when continue statement is executed, the statements following continue are skipped & causes the loop to be continued with the next iteration.
2. It can be used in switch statement to transfer the control outside switch	2. it can be used inside a for, while, & do-while loops.

3. example:-
for(i=1; i<=6; i++)
{
if(i==3){break;
printf("%d", i);
}}
#output:-
1 2
	3. example:-
	for(i=1; i<=5; i++)
	{
	if(i==3){continue;
	printf("%d", i);
	}
	#output:-
	1 2 4 5

4. Explain unconditional statements with example.

Ans:- Unconditional statements :- The statements that transfer the control from one place to another place in the program without any condition are called unconditional branch statements.

* there are 4 types:-

① goto ② break ③ continue ④ return.

→ refer 3rd answer

① goto :- using this statement, control can be transferred from one statement to the specified statement without any condition (unconditionally).

ARUN'S
PAGE NO.
DATE

Syntax:
goto label;

- Working:- execution of the goto statement causes the control to be transferred to the statement labelled with label.

- Example:-
```
#include<stdio.h>
void main()
{
  int sum=0, i=0;
top: if(i>10) goto end;
  sum=sum+i;
  i++;
  goto top;
end: printf("sum=%d\n", sum);
}
```

② Return:- It is used to specify a value or expression that a value or expression that a function should return to the calling function.

Working:- it terminate the execution of a function and return control to the calling function along with value or expression

Syntax:-
return expression;

example:
```
#include<stdio.h>
int add(int a, int b){
  int sum=a+b;
  return sum;
}
int main(){
```

```
int num1 = 5;
int num2 = 7;
int result = add(num1, num2);
printf("sum : %d\n", result);
return 0;
}
```

5. (a) difference b/w type conversion & type casting?

Ans.	Type conversion	Type casting
1	The C compiler converts the data type with higher rank.	The programmer can instruct the compiler to change the type of the operand from one data type to another data type.
2.	This process of conversion of data from lower rank to higher rank automatically by the C compiler is called type conversion (implicit type conversion).	(2) This forcible conversion from one data type to another data type is called type casting (explicit type conversion)

3.	ex:- int a = 10;
	float b = 10.0;
	float c = a + b;
	c = 20.0
		(3.) i = (int) 5.99 / (int) 2.4
		now it become 6/2 = 2,
		i = 2.

6. (a) define loop & difference b/w while loop & do-while loop.

Ans → Next page.

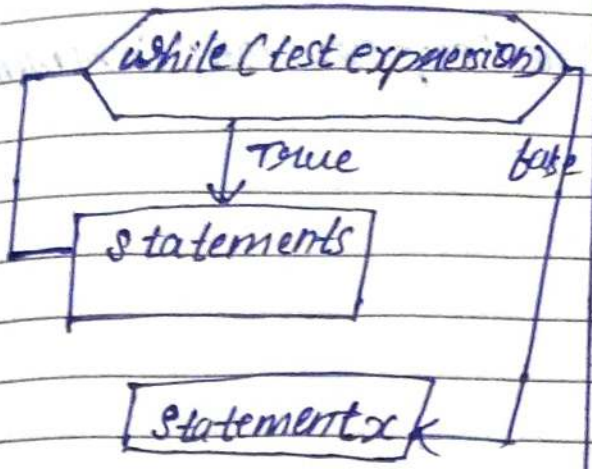
While loop	do-while loop
1. It is entry controlled loop. (top testing)	① It is exit controlled loop. (bottom testing)
2. It is pre-test loop	② It is post-test loop.

3. Syntax:-
while (expression)
{
statements;
}
	③ syntax:
	do {
	statements;
	} while (expression);

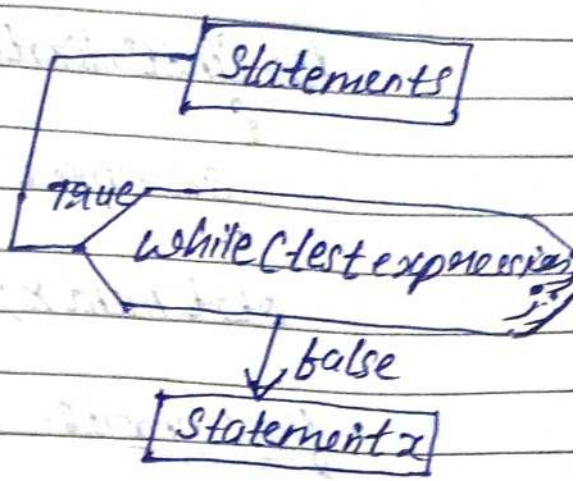
4. working principle:- If the expression is evaluated to true, then the statements within the body of loop are executed, if the expression is false, then the statements within body of the loop are not executed.	④ working principle:- On reaching do statatement, it proceeds to execute the statements within the body of the loop irrespective of the test expression. • If the expression is evaluated to true, then the statements within the body of loop are executed once again. • If the expression is false, then the statements within the body of the loop are not executed.
5. It does not ends with semicolon(;)	⑤ while statement ends with semicolon(;)

6. ex:-
int i=0, sum=0;
while (i<=n)
{ sum=sum+i;
i=i+1;
}
	⑥ ex:- int i=0, sum=0;
	do {
	sum=sum+i;
	i=i+1;
	} while (i<=n);

7. flowchart:-



(1) Flowchart :-



* Loops :-

* A set of statements may have to be repeatedly executed for a specified number of times or till a condition is satisfied.

* These statements are also called as repetitive or iterative statements.

Loop statements :-
- while loop
- for loop
- DO-while loop

7. explain for loop and Nested for loop with syntax and example.

Ans: for loop :- (counter-controlled loop) :-

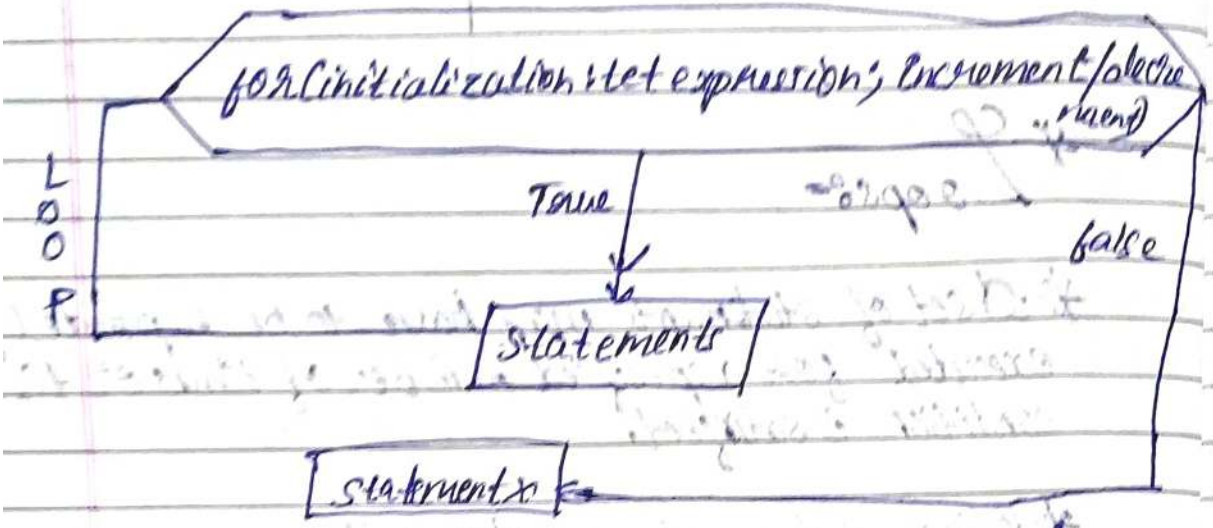
dif :- * It is a control statement using which the programmer can give instructions to the computer to execute a set of statements repeatedly for a specific number of times.

* it is also called as entry controlled loop or pretest loop.

- Syntax:-

for(initialization; test expression; increment/decrement)
{
statements;
}
statement x;

- flow chart:-



- Working principles-

* initialization:- In this section loop variable is initialized, like i=0, n=0, i=1, n=1.

* test expression:- The test expression may be a relational expression or logical expression or both, which is evaluated to true or false, depending on the value of the test expression, the body of the loop is executed.
- If the test expression is TRUE, then the body of the loop is executed.
- This process of execution of body of the loop is continued as long as the expression is TRUE.
- When the test expression becomes FALSE, execution.

of the statements contained in the body of the loop are skipped, thereby transferring the control to the statement x, which immediately follows the for loop.

* increment / decrement :- This section increments or decrements the loop variables after executing the body of the loop.

ex:- (program to find the factorial of a given number using for statement.

```
#include <stdio.h>
void main()
{
    int fact=1, i, n;
    printf("enter an integer:\n");
    scanf("%d", &n);
    for(i=1; i<=n; i++)
    {
        fact=fact*i;
    }
    printf("factorial of a given number is=%d\n", fact);
}
```

(9) Nested loop :-

- definition :- The loops that can be placed inside other loops called as nested loops.
  * it will work with any loops such as while, dowhile and for.
  * Nested loop is commonly used with the for loop because this is easiest to control.
- working :-
  * The 'inner for loop' can be used to control the number of times that particular set of statements will be executed.

ARUN'S
PAGE NO.
DATE / /

* The 'outer for loop' can be used to control the number of times that a whole loop is repeated.

ex:-

```
#include<stdio.h>
void main()
{
int i,j;
for(i=1;i<=5;i++)
{
printf("\n");
for(j=1;j<=i;j++)
{
printf("%d",j);
}
}
}
```

#output:-

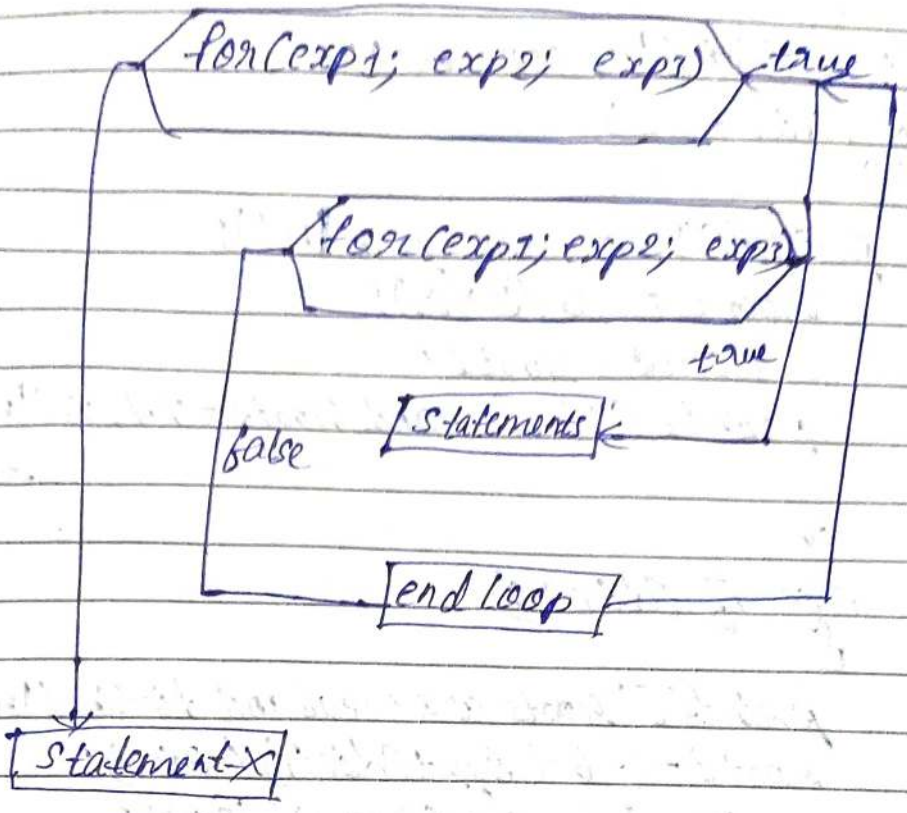
```
1
1 2
1 2 3
1 2 3 4
1 2 3 4.
```

Syntax:-

```
for(exp1;exp2;exp3)
{
for(exp1;exp2;exp3)
{
statement block;
}
}
```

ARUN'S
PAGE NO.
DATE / /

Flowchart:-



* Programs:-

1 (10) Develop a C program that takes three co-efficients (a, b, & c) of a quadratic equation; (ax^2+bx+c) as input and compute all possible roots and print them with appropriate messages.

Ans.
```
#include<stdio.h>
#include<math.h>
    void main()
    {
        float a, b, c, root1, root2, rpart, ipart, disc;
        printf("enter the 3 coefficients:\n");
        scanf("%f %f %f", &a, &b, &c);
        if (a*b*c)==0)
    }
```

```
printf("roots cannot be determined:\n");
exit(0);
}
disc = (b*b)-(4*a*c);
if (disc==0)
{
printf("roots are equal\n");
root1 = -b/2*a);
root2 = root1;
printf("root1 and root2 = %f, %f\n", root1,
                                    root2);
}
else if (disc>0)
{
printf("roots are real and distinct\n");
root1 = (-b + sqrt(disc))/2*a);
root2 = (-b - sqrt(disc))/(2*a);
printf("root1 = %f\n", root1);
printf("root2 = %f\n", root2);
}
else
{
printf("roots are complex\n");
rpart = -b/(2*a);
ipart = (sqrt(-disc))/(2*a);
printf("root1 = %f+i%f\n", rpart, ipart);
printf("root2 = %f-i%f\n", rpart, ipart);
}
}
```

#output:-

```
enter the 3 coefficients:-
1
2
1
roots are equal
root1 = -1.0000 root2 = -1.0000
```

ARUN'S
PAGE NO.
DATE / /

#output 2
enter the 3 coefficients:
2
3
2
roots are real & equal
root 1 = -0.5000
root 2 = -2.0000

#output 3
enter the 3 coefficients:
2
2
2
roots are complex
root 1 = -0.5000 + i0.8660
root 2 = -0.5000 + i0.8660

9. palindrome or not.

Ans.
```
#include <stdio.h>
#include <conio.h>
void main()
{
int n, digit, rev;
clrscr();
printf("enter the value of n\n");
scanf("%d", &n);
rev = 0;
do
{ digit = n % 10;
n = n / 10;
rev = (rev * 10) + digit;
} while (n != 0)
printf("reverse %d = %d, %d", n, rev);
if (n == rev) {
printf("no is a palindrome\n"); }
else { printf("no is not a palindrome\n");
}
}
```

3. GCD & LCM

Ans.
```
#include <stdio.h>
void main()
{
```

```
int m,n,a,b,r,gcd,lcm;
clrscr();
printf("enter the 2 integers\n");
scanf("%d %d",&m,&n);
a=m; b=n;
while(n!=0)
{
r=m%n;
m=n;
n=r
}
gcd=m;
lcm=(a*b)/gcd;
printf("gcd=%d",gcd);
printf("lcm=%d",lcm);
getch();
}
```

4. Sum of odd & even no :-

```
#include<stdio.h>
void main(){
int n,i,esum=0,osum=0;
clrscr();
printf("enter the value of n\n");
scanf("%d",&n);
for(i=1;i<=n;i++){
if(i%2==0){
esum+=i
}else{
osum+=i;
}
}
printf("Sum of odd no 1 to %d: %d\n",n,osum);
printf("Sum of even no 1 to %d: %d\n",n,esum);
}
```

ARUN'S
PAGE NO.
DATE / /

5. Fibonacci series or numbers:-

Ans:
```
#include <stdio.h>
void main(){
    int n, i;
    printf("enter the number of fibonacci numbers you want to generate: ");
    scanf("%d", &n);
    int fib1=0, fib2=1, fib3;
    printf("fibonacci numbers: %d %d", fib1, fib2);
    for(i=3; i<=n; i++){
        fib3 = fib1 + fib2;
        printf("%d", fib3);
        fib1 = fib2;
        fib2 = fib3;
    }
    printf("\n"); getch();
}
```

*Subscribe to our channel for next module answers.