Arithmetic ;

Q1 : Add two numbers 1 and 2 ;

1 + 2

3

Q2 : Subtruct two numbers 7 and 3 ;

7 - 3

4

Q3 : Multiply two numbers 2 and 3 ;

Note : For multiplication of two numbers we use sign * or by using space between the two multiplied numbers ;

2 * 3

6

2 × 3

6

Q4 : Divide two numbers 2 & 3 ;

Note : for division we use sign /

2 / 3

$\frac{2}{3}$

Note : to get the answer in fraction we use . with any number . or use N for get the answer in fraction ;

7. / 3

2.33333

N[7 / 3]

2.33333

Q5 : Factorial of number 5 ;

Note : for finding the factorial we can use sign ! or Factorial ;

5 !

120

Factorial [5]

120

Q6 : calculate 10 digits in 2 / 7 ;

Note : to get the number of digit by given question we can use the following form to get the number of digits of any number N[2 / 7, n] where n is number of required digits ;

```
N[2 / 7, 10]
```

0.2857142857

Note : this also be done by use = ing post version symbol which used for only one input. but this not give the required number of digit of required question; // N

```
2 / 7 // N
```

0.285714

Q7 : Calculate 10 digits in Pi;

```
N[Pi, 10]
```

3.141592654

Q8 : calculate the square root 5 and 4 power 3.1;

Note : to show or find the power of any number we use the hat operator ^

power;

```
4 ^ 3.1
```

73.5167

```
5 ^ (1 / 2)
```

$\sqrt{5}$

note : to show these square root to follow the BODMAS rule;

```
N[5 ^ (1 / 2), 10]
```

2.236067977

```
5 ^ (1 / 2) // N
```

2.23607

Note : to shoe the sqaure root we can use the command Sqrt[x] where x is any given number;

Q9 : to show the numbers in the form of sqaure roor;

```
Sqrt[Pi]
```

$\sqrt{\pi}$

note : to find the log or anti - log we use the command;

```
? Log
```

Log[z] gives the natural logarithm of z (logarithm to base e).
Log[b, z] gives the logarithm to base b. ≫

Q10 : calculate ln3 .1 and log [3, 9];

```
Log[3.1]
```

1.1314

```
Log[3, 9]
```

2

Q11 : calculate the log;

```
7^Log[7, 11]
```

11

```
Log[5, 5^19]
```

19

Note : to find the answer for trignometirx functions we use the built in fucntions, like Cos[30 Degree] or Cos[Pi / 6] where Degree = Pi / 180;

Q12 : find the answer ;

```
Cos[30 * Pi / 180]
```

$$\frac{\sqrt{3}}{2}$$

```
Cos[Pi / 6]
```

$$\frac{\sqrt{3}}{2}$$

```
Cos[30 * Degree]
```

$$\frac{\sqrt{3}}{2}$$

```
? Degree
```

Degree gives the number of radians in one degree. It has a numerical value of $\frac{\pi}{180}$. ≫

Q13 : find inverse trignometric functions in degree;

Note : we use the command ArcSin[x] where x is any given value;

```
ArcCos[1 / 2] / Degree
```

$$\frac{\pi}{3\,°}$$

Q14 : find inverse trignometric functions in degree and in decimal points;

```
N[ArcCos[1 / 2] / Degree]
```

60.

```
N[ArcSin[1 / 2] / Degree]
```

30.

```
ArcCos[1 / 2] / Degree // N
```

60.

Topic : Order of operations;

note : to BODMAS rules;

```
Pi^(2 / 3)
```

$\pi^{2/3}$

```
5^(3 Ie)
```

$5^{3\,Ie}$

```
I^((3/2) + Pi)
```

$i^{\frac{3}{2}+\pi}$

```
Pi^(3/2 I)
```

$\pi^{\frac{3 i}{2}}$

```
12 Pi * I / (((Pi + 2)^I / (4 Pi)) + 5 I)
```

$$\frac{12\, i\, \pi}{5\, i + \frac{(2+\pi)^{i}}{4\,\pi}}$$

```
-Sin[2 + Pi * I] / ((Pi + 1)^2 / (8 e) + 2 e^3)^(3/2)
```

$$-\frac{\mathrm{Sin}[2 + i\,\pi]}{\left(2\, e^{3} + \frac{(1+\pi)^{2}}{8\, e}\right)^{3/2}}$$

Topic : Expression ;

```
x = 2
```

2

Q15 : what is output ;

```
x = 2; y = 3; ab = 5
```

5

```
x = 2; y = 3; xy = 5
```

5

```
x = 2; y = 3; x * y
```

6

```
x = 2
y = 4
```

2

4

```
x = 2; y = 3; x * y; Print[x * y]
```

6

```
x = 2; y = 3; x * y; x = 3; Print[x * y]
```

9

```
x = 2; y = 3; x * y; x = 4; Print[x]
```

4

Topic : Loops ;

Note : for loops we use the command Do and For and to display the values of the lop we use the command Print[x] where x is any number whose we want to print the value ;

```
?Do
```

Do[*expr*, {i_{max}}] evaluates *expr* i_{max} times.
Do[*expr*, {*i*, i_{max}}] evaluates *expr* with the variable *i* successively taking on the values 1 through i_{max} (in steps of 1).
Do[*expr*, {*i*, i_{min}, i_{max}}] starts with $i = i_{min}$.
Do[*expr*, {*i*, i_{min}, i_{max}, *di*}] uses steps *di*.
Do[*expr*, {*i*, {i_1, i_2, …}}] uses the successive values i_1, i_2, ….
Do[*expr*, {*i*, i_{min}, i_{max}}, {*j*, j_{min}, j_{max}}, …] evaluates *expr* looping over different values of *j*, etc. for each *i*. ≫

```
?For
```

For[*start*, *test*, *incr*, *body*] executes *start*, then repeatedly evaluates *body* and *incr* until *test* fails to give True. ≫

```
Do[2 i, {i, 1, 4}] // in this example we cant disply
   the number of this loop because we not use commadn Print[x] here;

Do[Print[2 i], {i, 1, 4}]
```

```
2

4

6

8
```

```
Q16 : generate all elements of the sequence a1 = 2.1, a2 = 6, a3 = 5,
and the sequence is that a1 = i^2 + i for i = 1, 2, 3,  ... .., 12;

a[1] = 2.1; a[2] = 6; a[3] = 5; Do[Print[i^2 + i], {i, 1, 12}]
```

```
2

6

12

20

30

42

56

72

90

110

132

156
```

```
Q17 : generate all elements of the sequence a1 = 2.1,
a2 = 6, a3 = 5, and the sequence is that a[i] for i = 1, 2, 3;

a[1] = 2.1; a[2] = 6; a[3] = 5; Do[Print[a[i]], {i, 1, 3}]
```

2.1

6

5

Q18 : generate all elements of the sequence given by the starter formula a1 = 2 and the recursion formula a[i + 1] = 2 ai - 3 for i = 1, 2, 3, ... .12;

```
a[1] = 2
Do[Print[2 a[i] - 3], {i, 1, 12}]
```

2

1

-1

-5

-13

-29

-61

-125

-253

-509

-1021

-2045

-4093

this question also be done by;

```
a[1] = 2
a[i + 1] = 2 a[i] - 3; Do[Print[a[i + 1]], {i, 1, 12}]
```

2

1

-1

-5

-13

-29

-61

-125

-253

-509

-1021

-2045

-4093

Topic : Genrating a list and a list of lists; rows, columns and matrices;

Note : to generate a list of numbers we use the commnad Table,
which represent each number separated by commas in {}; you can check the command Table and Do;

```
? Do
```

Do[*expr*, {i_{max}}] evaluates *expr* i_{max} times.
Do[*expr*, {*i*, i_{max}}] evaluates *expr* with the variable *i* successively taking on the values 1 through i_{max} (in steps of 1).
Do[*expr*, {*i*, i_{min}, i_{max}}] starts with $i = i_{min}$.
Do[*expr*, {*i*, i_{min}, i_{max}, *di*}] uses steps *di*.
Do[*expr*, {*i*, {i_1, i_2, …}}] uses the successive values i_1, i_2, ….
Do[*expr*, {*i*, i_{min}, i_{max}}, {*j*, j_{min}, j_{max}}, …] evaluates *expr* looping over different values of *j*, etc. for each *i*. ≫

```
? Table
```

Table[*expr*, {i_{max}}] generates a list of i_{max} copies of *expr*.
Table[*expr*, {*i*, i_{max}}] generates a list of the values of *expr* when *i* runs from 1 to i_{max}.
Table[*expr*, {*i*, i_{min}, i_{max}}] starts with $i = i_{min}$.
Table[*expr*, {*i*, i_{min}, i_{max}, *di*}] uses steps *di*.
Table[*expr*, {*i*, {i_1, i_2, …}}] uses the successive values i_1, i_2, ….
Table[*expr*, {*i*, i_{min}, i_{max}}, {*j*, j_{min}, j_{max}}, …] gives a nested list. The list associated with *i* is outermost. ≫

you can see that comnands are same in patterns but the more efiicient for lists are Table command;

Q19 : generate a list containing 2.1, 6 and 5;

```
{2.1, 6, 5}
```

```
{2.1, 6, 5}
```

Q20 : generate a list containing all elements of the sequence a[i] = i^2 + i for i = 1, 2, ... .12;

```
Table[i^2 + 1, {i, 1, 12}]
```

```
{2, 5, 10, 17, 26, 37, 50, 65, 82, 101, 122, 145}
```

Q21 : generate a list of all elements of the sequence given by the starter formula a1 = 2 and the recursion formula a[i + 1] = 2 ai - 3 for i = 1, 2, 3, ... .12;

note : to add the starter formula as the output value we use the another command with tyhe table are Append or Prepend;

```
a[1] = 2; Prepend[Table[2 a[i] - 3, {i, 1, 12}], a[1]]
```

```
{2, 1, -1, -5, -13, -29, -61, -125, -253, -509, -1021, -2045, -4093}
```

Note : a matrix is a list of lists, i.e. a list of rows with each row being a list as above;

Q22 : type in the matrix and display normally;

to display the list in the form of matrix we use the commands TraditionalForm[], MatrixForm[], both commands yield same result;

```
TraditionalForm[{{-1, 6.7}, {-Pi, I}}]
```

$$\begin{pmatrix} -1 & 6.7 \\ -\pi & i \end{pmatrix}$$

```
MatrixForm[{{-1, 6.7}, {-Pi, I}}]
```

$$\begin{pmatrix} -1 & 6.7 \\ -\pi & i \end{pmatrix}$$

multiplying rows, columns and matrices;

Q23 : multiply a column with a row;

Note : In mathematica,
if you have assigned the list say r (row) and assigned a list to c (coloumn),
for this multiplication you have to type in r.c after these multiplication;

```
r = {{5}, {0}, {-1}}; c = {{2, 7, 3}}; TraditionalForm[r.c]
```

$$\begin{pmatrix} 10 & 35 & 15 \\ 0 & 0 & 0 \\ -2 & -7 & -3 \end{pmatrix}$$

Q24 : multiply the matrix with column and row;

Note : when we multiply a matrix with another list,
the list appearing to the left of a dot becomes a row (multiply the matrix with row) r.m and the list appearing to the right of a dot becomes a column (multiply the matrix with column) m.c;

```
m = {{1, 0, Pi}, {8, 3, 2}, {I, -5, 1.5}}; r = {{2, 7, 3}}; c = {{5}, {0}, {-1}};
"matirx with row"
TraditionalForm[r.m]
"matrix with column"
TraditionalForm[m.c]
```

matirx with row

$$(\, 58. + 3.\, i \quad 6. + 0.\, i \quad 24.7832 + 0.\, i \,)$$

matrix with column

$$\begin{pmatrix} 1.85841 + 0.\, i \\ 38. + 0.\, i \\ -1.5 + 5.\, i \end{pmatrix}$$

matrices with rules for elements;

Q25 : generate a matrix containing all the elements given by a[i][j] =
(i - 1)^2 j^3 s[i][j] for i, j = 1, 2,
3 and display it normally where s = [i][j] is kroneckerDelta;

to solve this question we use the command Table and put in the commnad TraditionalForm (first make the list and then display it normally);

```
?KroneckerDelta
```

KroneckerDelta[n_1, n_2, …] gives the Kronecker delta $\delta_{n_1 n_2 \ldots}$, equal to 1 if all the n_i are equal, and 0 otherwise. ≫

```
A = Table[(i - 1)^2 * j^3 KroneckerDelta[i, j], {i, 1, 3}, {j, 1, 3}]; TraditionalForm[A]
```

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 108 \end{pmatrix}$$

Q26 : generate a matrix containing all the elements given by a[i][j] = i^2 SinjPi for i,
j = 1, 2, 3 and display it normally;

```
T = Table[i^2 Sin[j * Pi], {i, 1, 3}, {j, 1, 3}]; TraditionalForm[T]
```

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Q27 : Generate an identity matrix of 5 cross 5;

to generate the n cross identity matrix we use the commnad IdentityMatrix[n] where n is n cross n required number of rows and columns;

```
? IdentityMatrix
```

IdentityMatrix[n] gives the $n \times n$ identity matrix. ≫

```
iden = IdentityMatrix[3]; TraditionalForm[iden]
```

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```
Q : geometric sequence;
a[1] = 4
r = 3; Do[Print[a[1] * r^i], {i, 1, 9}]
```

4

12

36

108

324

972

2916

8748

26 244

78 732

```
Q : Arithmetic sequence;
a[1] = 3; r = 2
Do[Print[a[i + 1] = a[i] + r], {i, 1, 9}]
```

2

5

7

9

11

13

15

17

19

21

piecewise function;

```
f[x_] := 2 / x /; x < 1
f[x_] := x /; 1 <= x <= 2
f[x_] := x^2 /; x > 2
x = 2.5; f[x]
```

6.25

```
f[x_] := x^2; f[t]
```

t^2

```
f[x_, y_] := x^y; f[b, 2]
```

```
m1 = 3; m2 = 5; x1 = 2 * 10^-2; xc := (m1 * x1 + m2 * x2) / (m1 + m2)
x2 = 4 * 10^-2;
xc
"after repeat"
x2 = 3 * 10^-2;
xc
```

$\frac{13}{400}$

after repeat

$\frac{21}{800}$

topic : symbolic representation;

```
Expand[(a + b)^3]
```

$a^3 + 3 a^2 b + 3 a b^2 + b^3$

Q : by using coloumb ' s law total flux through surface of radius r with a charge q at its centre in terms of permitivity z;

```
F = 1 / (4 Pi z) (q1 q / r^2); ef = F / q1; fl = ef 4 Pi r^2 Cos[0]
```

$\frac{q}{z}$

```
f[x_] := x^2; f[x]
```

x^2

```
f[x_] := x^2; f[3]
```

9

```
? _
```

_ or Blank[] is a pattern object that can stand for any *Mathematica* expression.
_*h* or Blank[*h*] can stand for any expression with head *h*. ≫

```
? /;
```

patt /; *test* is a pattern which matches only if the evaluation of *test* yields True.
lhs :> *rhs* /; *test* represents a rule which applies only if the evaluation of *test* yields True.
lhs := *rhs* /; *test* is a definition to be used only if *test* yields True. ≫

```
? If
```

If[*condition*, *t*, *f*] gives *t* if *condition* evaluates to True, and *f* if it evaluates to False.
If[*condition*, *t*, *f*, *u*] gives *u* if *condition* evaluates to neither True nor False. ≫

```
discussion;
```

```
x = 2; f = x^2; x = 3; f
```

4

```
x = 2; f := x^2; x = 3; f
```

9

```
c1 = 0.5 * 10^-6; c2 = 1.3 * 10^-6; ct = 1 / c1 + 1 / c2; 1 / ct
```

3.61111×10^{-7}

```
Q : Total number of electrons passing through circuit in 2 minutes if current is 3 mA;
we use formulas is n (total number of elctrons) = I * t / e;
```

```
t = 120; i = 3 * 10^-3; e = 1.67 * 10^-19; n = i * t / e
```

2.15569×10^{18}

```
functions;
```

```
? _
```

_ or Blank[] is a pattern object that can stand for any *Mathematica* expression.
_*h* or Blank[*h*] can stand for any expression with head *h*. ≫

```
? /;
```

patt /; *test* is a pattern which matches only if the evaluation of *test* yields True.
lhs :> *rhs* /; *test* represents a rule which applies only if the evaluation of *test* yields True.
lhs := *rhs* /; *test* is a definition to be used only if *test* yields True. ≫

```
? If
```

If[*condition*, *t*, *f*] gives *t* if *condition* evaluates to True, and *f* if it evaluates to False.
If[*condition*, *t*, *f*, *u*] gives *u* if *condition* evaluates to neither True nor False. ≫

```
f[x_] := x^2; f[t]
```

t^2

```
f[x_, y_] := x^y; f[b, 2]
```

b^2

```
piecewise function;
```

```
f[x_] := 2 / x /; x < 1
f[x_] := x /; 1 <= x <= 2
f[x_] := x^2 /; x > 2
x = 2.5; f[x]
```

6.25

```
f[x_] := 2 / x /; x < 1
f[x_] := x /; 1 <= x <= 2
f[x_] := x^2 /; x > 2
f[2.5]
```

6.25

```
Topic Absoulute;
```

```
? Abs
```

Abs[*z*] gives the absolute value of the real or complex number *z*. >>

```
f[x_] := x /; x >= 0
f[x_] := -x /; x < 0
f[-2] - Abs[-2]
```

0

```
? Simplify
```

Simplify[*expr*] performs a sequence of algebraic
and other transformations on *expr*, and returns the simplest form it finds.
Simplify[*expr*, *assum*] does simplification using assumptions. >>

```
calculate square root x square if x is negative;
```

```
Simplify[Sqrt[x^2], x < 0]
```

$-x$

```
topic : limit, differentiation and replacment rule;
```

```
? Limit
```

Limit[*expr*, $x \to x_0$] finds the limiting value of *expr* when x approaches x_0. >>

```
?D
```

D[f, x] gives the partial derivative $\partial f/\partial x$.
D[f, {x, n}] gives the multiple derivative $\partial^n f/\partial x^n$.
D[f, x, y, …] differentiates f successively with respect to x, y, ….
D[f, {{x_1, x_2, …}}] for a scalar f gives the vector derivative ($\partial f/\partial x_1$, $\partial f/\partial x_2$, …). ≫

```
Q1 : Differentiate sin[x] by using the defination and built in commnad differentiate ;

f[x_] := Sin[x];
"by defination"
Limit[(f[x + h] - f[x]) / h, h → 0]
"by commnad"
D[f[x], x]
```

```
by defination

Cos[x]

by commnad

Cos[x]
```

```
f[x_] := -7 x^3 + 5 x^2 + 11 x - 9;

"by definition"
Limit[(f[x + h] - f[x]) / h, h → 0]
"by commnad"
D[f[x], x]
```

by definition

$11 + 10 x - 21 x^2$

by commnad

$11 + 10 x - 21 x^2$

```
Q3 : Diffierentiate lnx at x = 1;

D[Log[x], x]
```

$\frac{1}{x}$

```
D[Log[x], x] /. x → 1
```

1

```
Q4 : Find the expression as well as SI values of angular velocity,
angular accelration, tangentional velocity, tangentional accelration,
and radial accelration for given equation, with r = 2 (meter) and t = 3 (sec);
```

```
th = 4 t^3 - 2 t^2 + 5 t - 9;
"w expression"
w = D[th, t]
"w value"
w /. t → 3
"alpha expression"
alpha = D[w, t]
"alpha value"
alpha /. t → 3
"tang speed expression"
vt = r w
"tang speed value"
vt /. r -> 2 /. t → 3
"tang alpha expression"
at = r alpha
"tang alpha value"
at /. r → 2 /. t → 3
"radial accel expression"
ar = vt^2 / r
"radial accel value"
ar /. r → 2 /. t → 3
```

w expression

$5 - 4 t + 12 t^2$

w value

101

alpha expression

$-4 + 24 t$

alpha value

68

tang speed expression

$r \left(5 - 4 t + 12 t^2\right)$

tang speed value

202

tang alpha expression

$r (-4 + 24 t)$

tang alpha value

136

radial accel expression

$r \left(5 - 4 t + 12 t^2\right)^2$

radial accel value

20 402

```
Q5 : find limit;

Limit[x / x, x → 0]

1

Q6 : Q : Use lagrane and Euler lagrange ' s ;

note :  to solve this question we use the command ?D and ?';

L = 1 / 2 m * x'[t]^2 - V[x[t]]; D[L, x[t]] - D[D[L, x'[t]], t] /. x''[t] → a /. -V'[x[t]] → F

F - a m


?D
```

D[f, x] gives the partial derivative $\partial f/\partial x$.
D[f, {x, n}] gives the multiple derivative $\partial^n f/\partial x^n$.
D[f, x, y, …] differentiates f successively with respect to x, y, ….
D[f, {{x_1, x_2, …}}] for a scalar f gives the vector derivative ($\partial f/\partial x_1$, $\partial f/\partial x_2$, …). ≫

```
note : to solve the question 1 we use the 2 nd property
    of Do command D[f, {x, n}] gives the multiple derivative ∂^n f / ∂x^n;

Q : Demonstrate that both x =
   ACos[wt] + BSin[wt] and x = cSin[wt] are solution of D[x, {t, 2}] + w^2 * t;

x = c * Sin[w * t + phi]; D[x, {t, 2}] + x * (w^2)

0

"there was a small mistake in following program do yourself";

x = A * Cos[w * t] + B * Sin[w * t]; D[x, {t, 2}] + x * (w^2)

-A w^2 Cos[t w] - B w^2 Sin[t w] + w^2 (A Cos[t w] + B Sin[t w])

topic : graphing function;

?D
```

D[f, x] gives the partial derivative $\partial f/\partial x$.
D[f, {x, n}] gives the multiple derivative $\partial^n f/\partial x^n$.
D[f, x, y, …] differentiates f successively with respect to x, y, ….
D[f, {{x_1, x_2, …}}] for a scalar f gives the vector derivative ($\partial f/\partial x_1$, $\partial f/\partial x_2$, …). ≫

```
Q1 : Demonstarte psi = ACos[kx - w * t + phi] is a solution
    of classical wave equation and make its graph and simulate its graph;
```

```
w = k * v; psi = ACos[k * x - w * t + phi]; D[psi, {x, 2}] - 1 / v^2 D[psi, {t, 2}]
```

0

note : to solve the question 1 we use the 2 nd property of Do command D[f, {x, n}] gives the multiple derivative $\partial^n f / \partial x^n$;

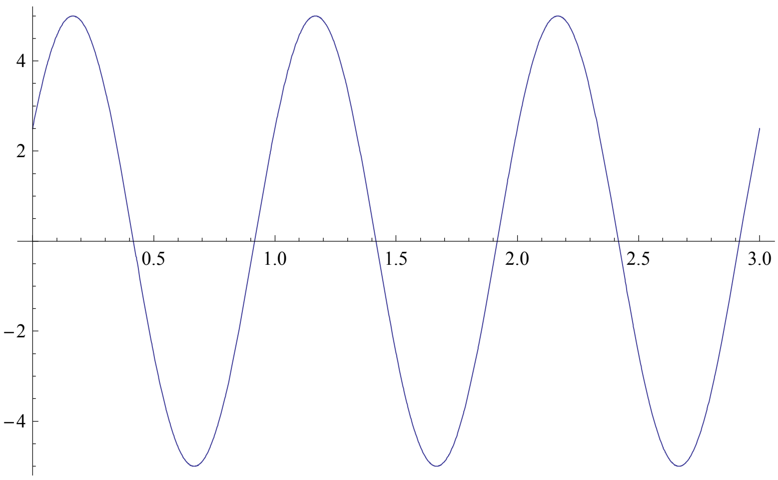
Q2 : graph the position, velocity and accelration for a one - dimensional SHO for the interval 0 ≤ t ≤ 3. take the amplitude of the motion to be 5 units, angular frequency 2 Pi radians adn the phase constant phi to be 30 Degree;

```
c = 5; w = 2 Pi; phi = 30 Degree;
"position"
x = c * Sin[w * t + phi]
"position plot"
Plot[x, {t, 0, 3}]
"velocity"
v = D[x, t]
"velocity plot"
Plot[a, {t, 0, 3}]
"accelration"
a = D[x, {t, 2}]
"accelration plot"
Plot[a, {t, 0, 3}]
```

"position"

5 Sin[30 ° + 2 π t]

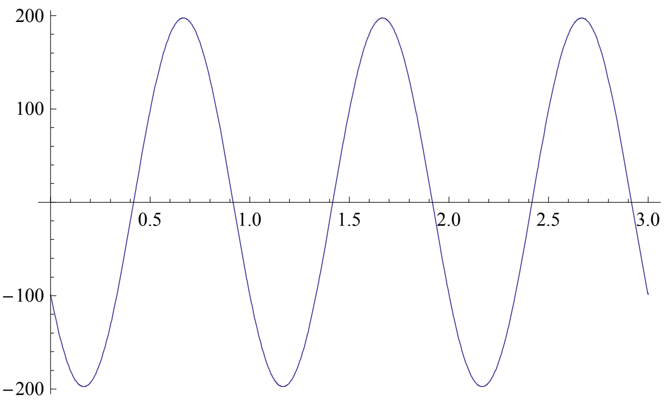
"position plot"



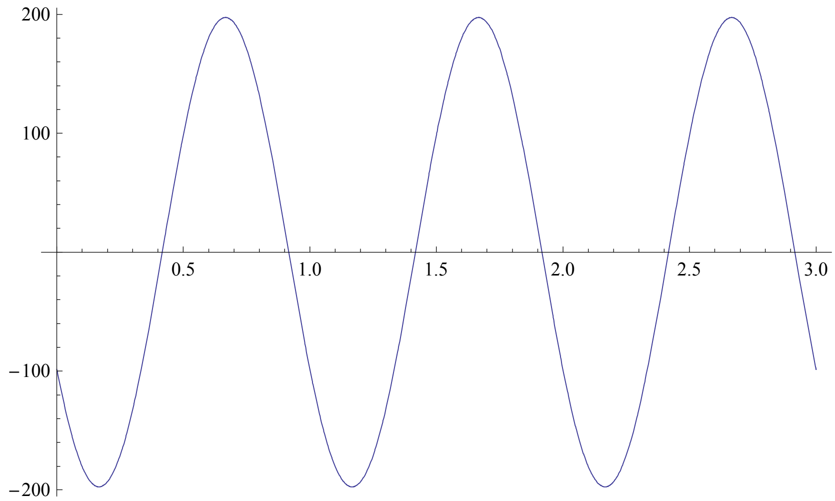
"velocity"

10 π Cos[30 ° + 2 π t]

"velocity plot"




```
"accelration"
-20 π^2 Sin[30 ° + 2 π t]
"accelration plot"
```



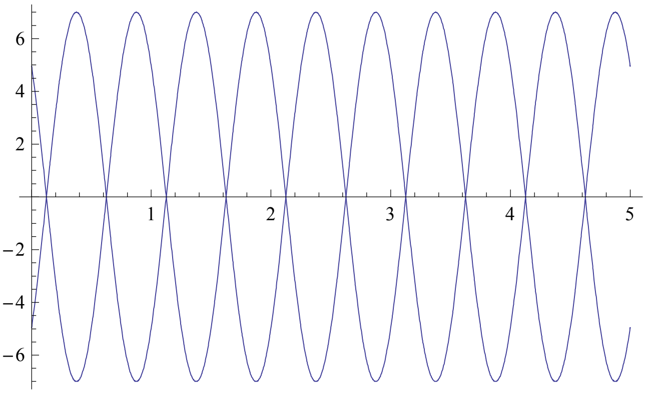

```
? Plot
```

Plot[f, {x, x_{min}, x_{max}}] generates a plot of f as a function of x from x_{min} to x_{max}.
Plot[{f_1, f_2, …}, {x, x_{min}, x_{max}}] plots several functions f_i. ≫

```
A = 5; w = 2 Pi; phi = 30 Degree; Animate[Plot[ACos[k * x - w * t + phi], {x, 0, 3}], {t, 0, 3}]
```

```
Q4 : for 0 ≤ x ≤ 5 meters and a wave with amplitude 7 units,  wave number 2 Pi radians / meter,
angular frequuency Pi radian / sec and phase constant 45 Degree for t =
  1 and t = 2 seconds. Disply bothe the graphs together;
```

```
t = 1; A = 7; k = 2 Pi; w = Pi; phi = 45 Degree; g1 = Plot[A * Cos[k * x - w * t + phi], {x, 0, 5}];
t = 2; g2 = Plot[A * Cos[k * x - w * t + phi], {x, 0, 5}]; Show[g1, g2]
```



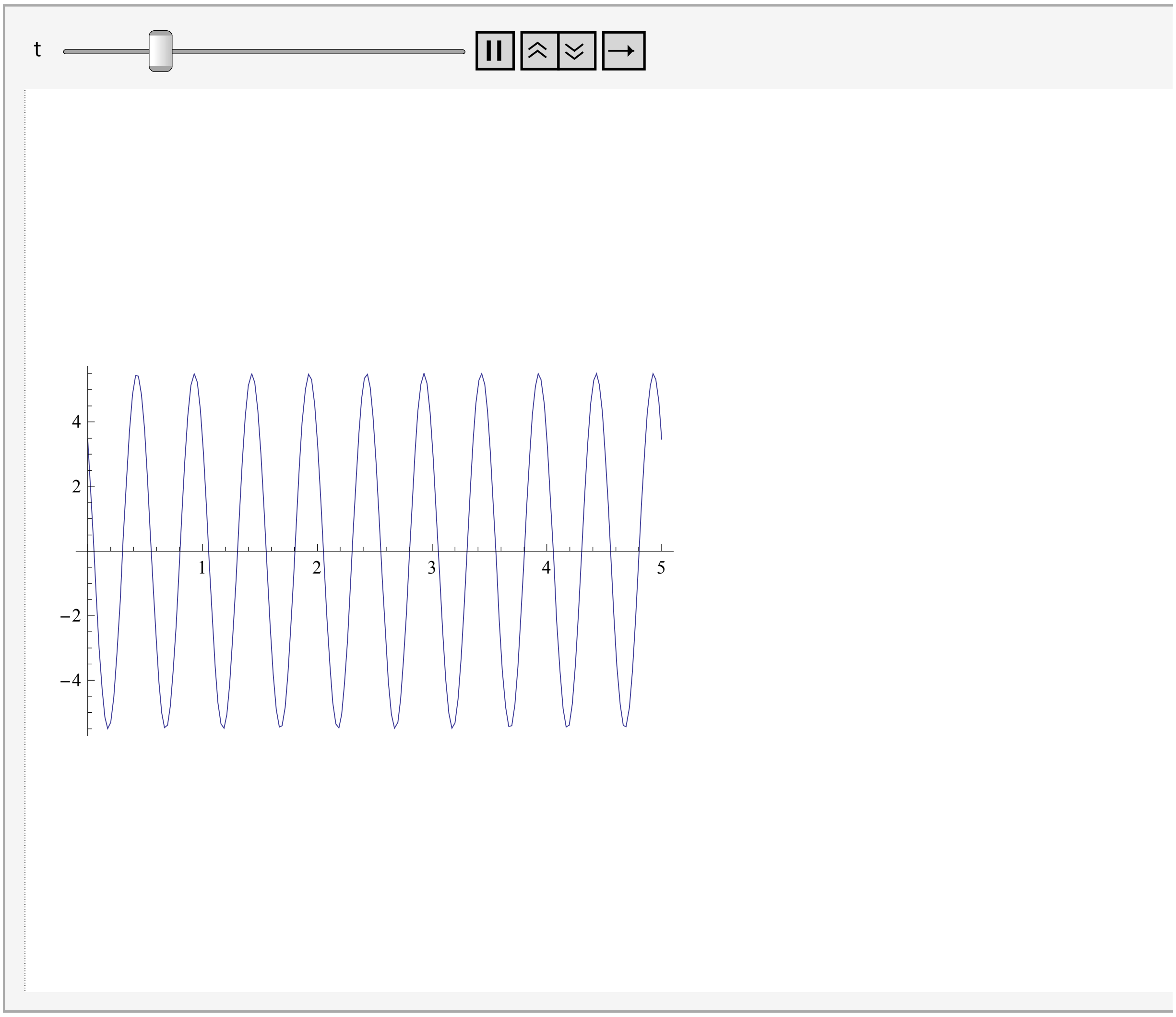

```
note : to show the both graphs we use the command ? Show;

? Show
```

Show[*graphics*, *options*] shows graphics with the specified options added.
Show[g_1, g_2, …] shows several graphics combined. ≫

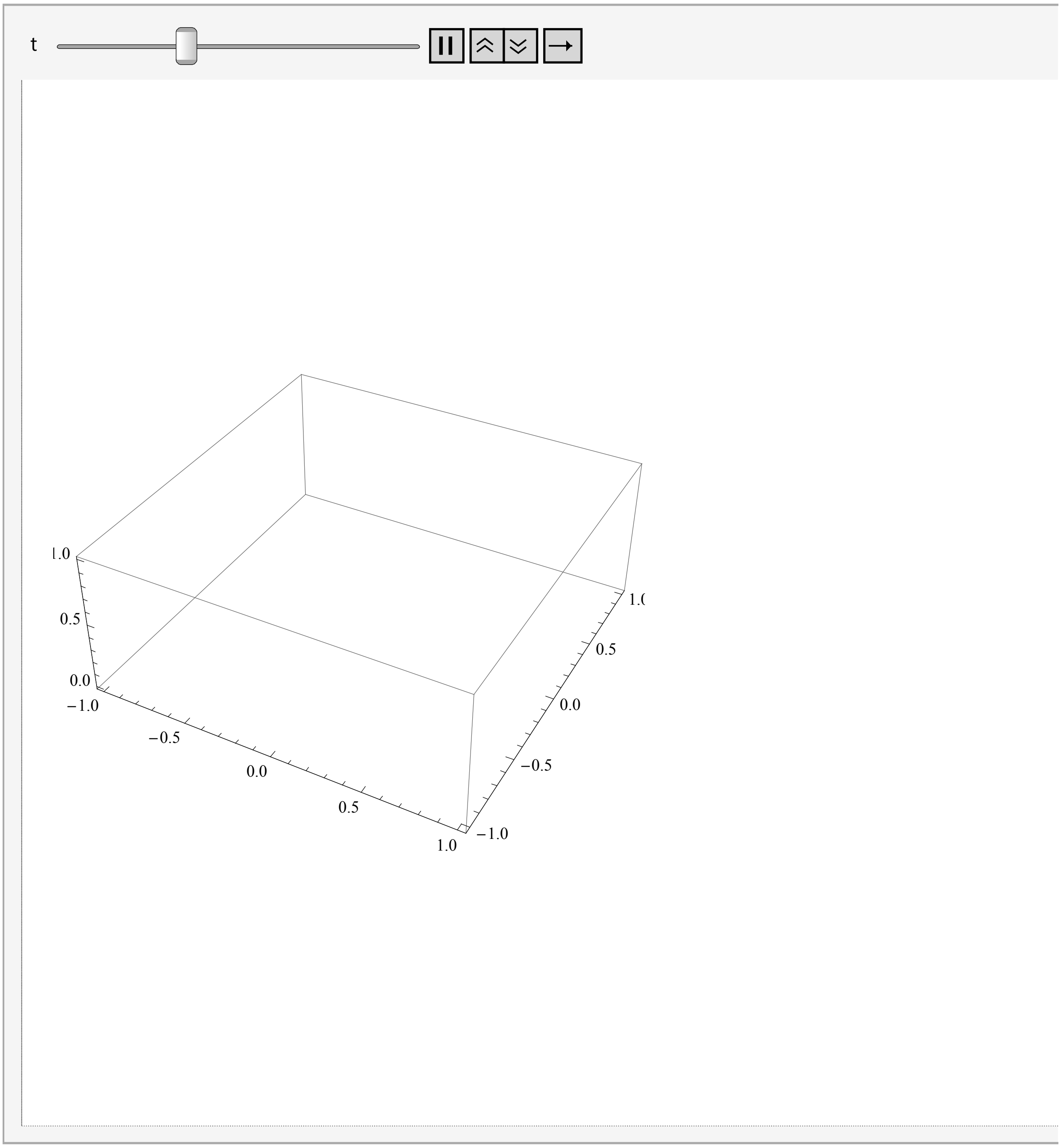
```
topic : Simulations;

to simulate the graph we use the command Animate;
```

In[1]:=
```
w = Pi; k = 4 Pi; A = 5.5; phi = 0;
Animate[Plot[psi = A*Cos[k*x - w*t + phi], {x, 0, 5}], {t, 1, 5}]
```

Out[1]=

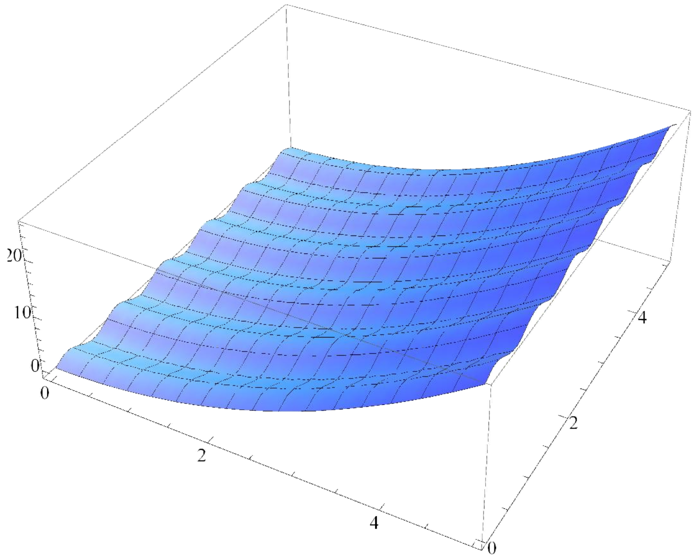


```
w = 3 Pi; kx = 2; ky = 3; A = 7; phi = 45;
Animate[Plot3D[Sin[kx * x + ky * y - w * t + phi], {x, 1, 3}, {y, 0, 4}], {t, 0, 2}]
```



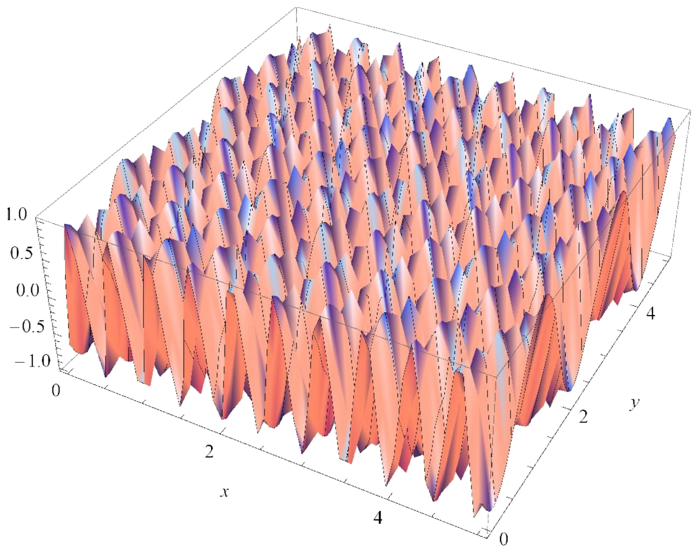

```
Q : for 0 ≤ x <= 5;  0 ≤ y <= 5; graph x62 + Sin7y;
```

```
Plot3D[x^2 + Sin[7 y], {x, 0, 5}, {y, 0, 5}]
```



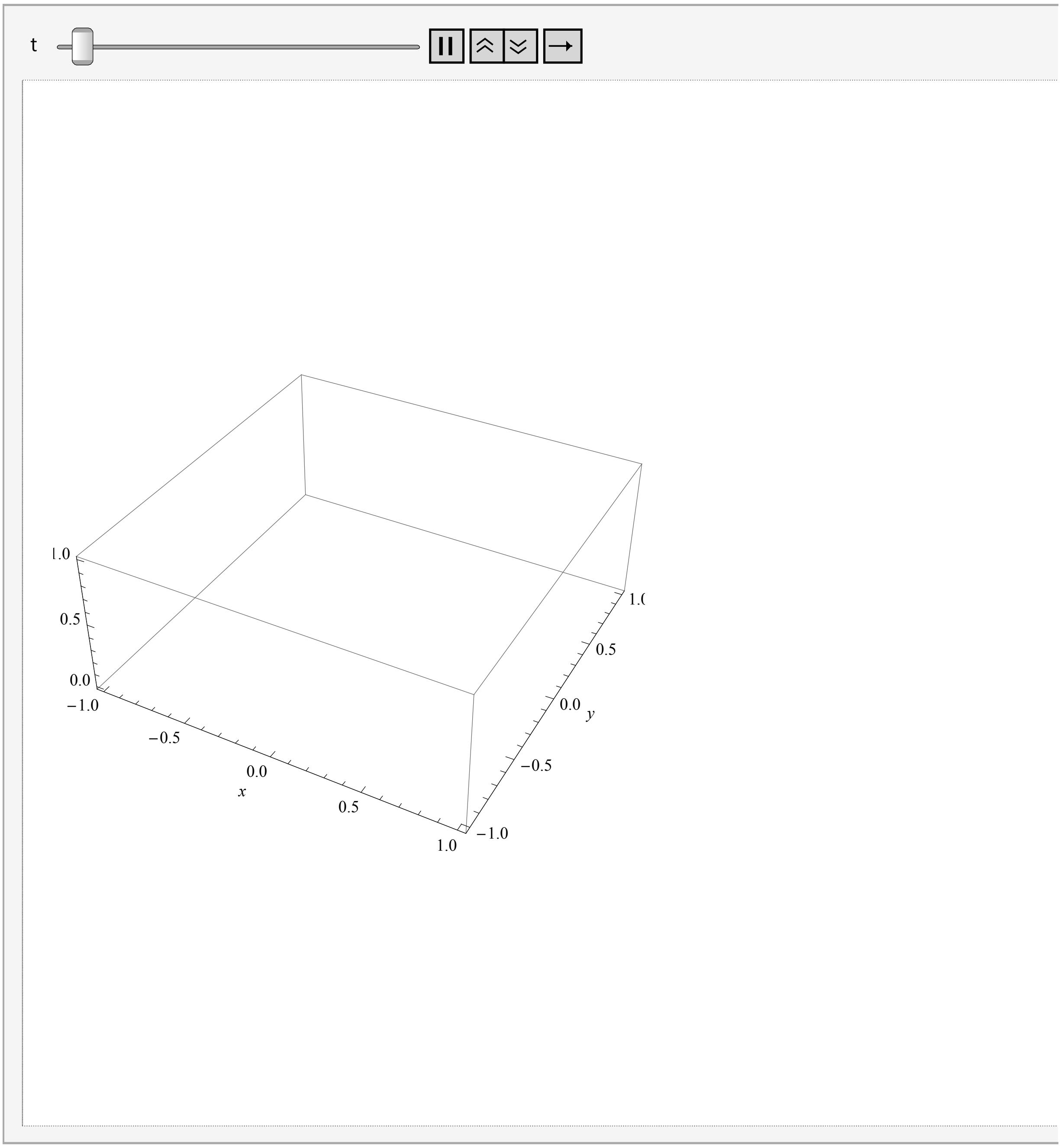
**Q : for 0 ≤ x <= 5; 0 ≤ y <= 5; graph a wave function with amplitude 7 units,
wave vector - 4 Pii + Pij radians / meter,
angular frequency 3 Pi radians / s and phase constant 45 degree. label x and y axis.**

```
Plot3D[Sin[-4 Pi * x + Pi * y - 3 Pi * 1 + 45 Degree], {x, 0, 5}, {y, 0, 5}, AxesLabel → Automatic]
```



Q :

```
w = 8 Pi; kx = 2; ky = 3; A = 5; phi = 0;
Animate[Plot3D[Sin[kx * x + ky * y - w * t + phi], {x, 0, 5}, {y, 0, 5},
  PlotPoints → {10}, AxesLabel → Automatic], {t, 1, 5}]
```

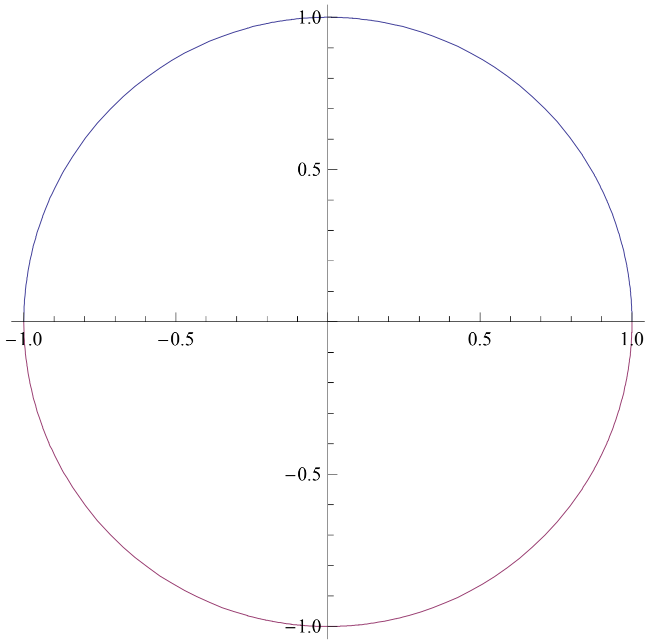



```
Q1 : Demonstarte psi = ACos[kx - w * t + phi] is a solution
   of classical wave equation and make its graph and simulate its graph;

w = k * v; psi = ACos[k * x - w * t + phi]; D[psi, {x, 2}] - 1 / v^2 D[psi, {t, 2}]
```

0

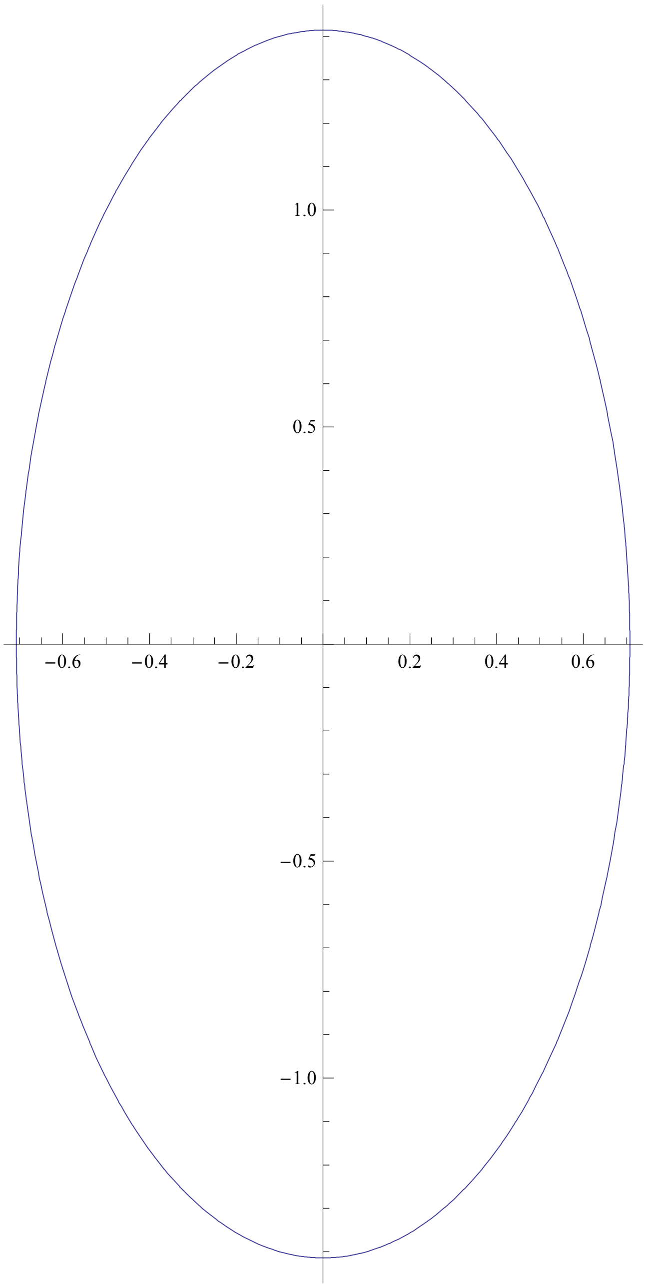
```
topic : parametric equatiuons and plots;

Q : manually solve for y and graph x^2 + y^2 = 1;
```

```
Plot[{Sqrt[1 - x^2], -Sqrt[1 - x^2]}, {x, -1, 1}, AspectRatio → Automatic]
```



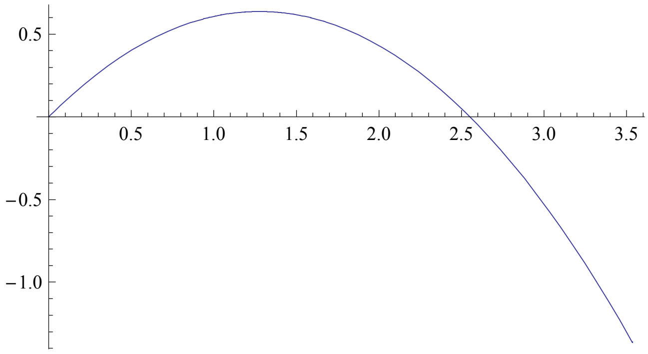

```
Q : use a parameter and parametric equations to graph 2 x^2 + 1 / 2 y^2 = 1;
```

```
ParametricPlot[{1 / Sqrt[2] Cos[t], Sqrt[2] Sin[t]}, {t, 0, 2 Pi}]
```




```
Q : parametric plot for projectile motion;
```

```
v = 5; thi = 45 Degree; g = 9.8;
ParametricPlot[{v*t*Cos[thi], v*t*Sin[thi] - 1/2 g*t^2}, {t, 0, 1}]
```




```
topic : contourPlot;
```

```
? ContourPlot
```

ContourPlot[f, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}] generates a contour plot of f as a function of x and y.
ContourPlot[f == g, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}] plots contour lines for which $f = g$.
ContourPlot[{f_1 == g_1, f_2 == g_2, …}, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}] plots several contour lines. »

```
Note : ContourPlot commnad is exactly is same as that
   of Plot3D commnad which is used for two variables. to do the questions
   for ContourPlot we just replace the Plot3D comad with the ContourPlot;
```

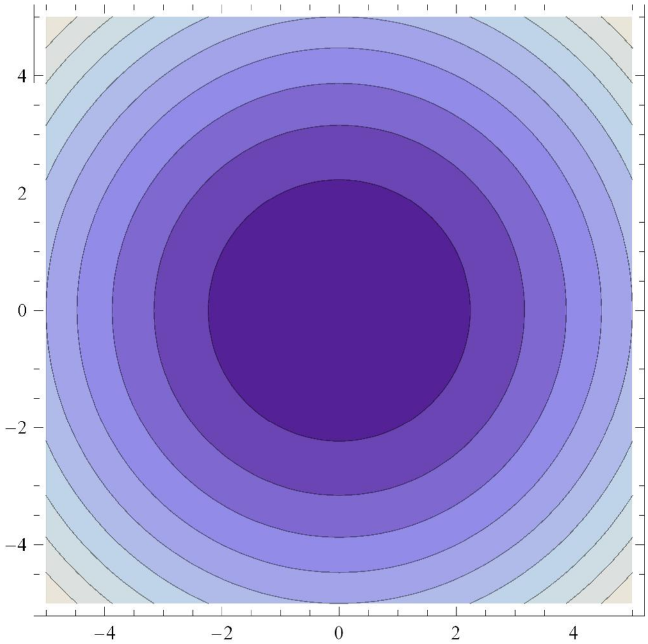
```
? Plot3D
```

Plot3D[f, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}] generates a three-dimensional plot of f as a function of x and y.
Plot3D[{f_1, f_2, …}, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}] plots several functions. »

```
Q1 : draw a regions / curves in {x, -5, 5}, {y, -5, 5} oor the surface x^2 + y^2;
```

```
Hint : to solve this question we use the first help from the ContourPlot commnad;
```

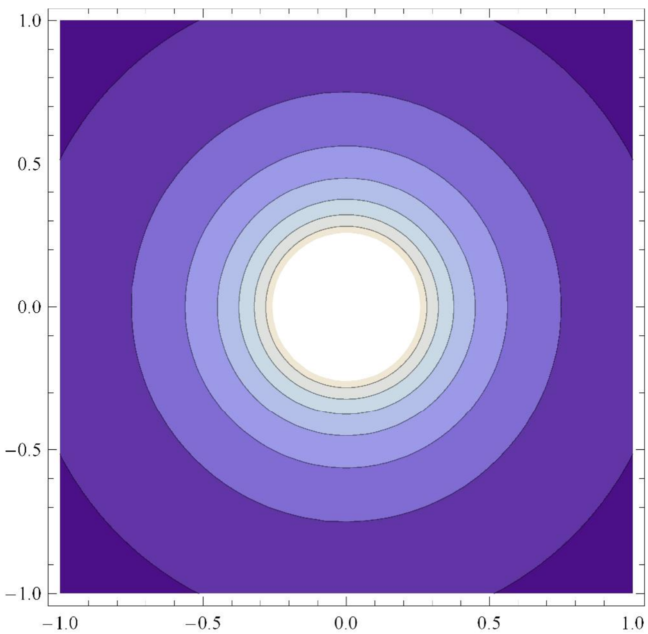
```
ContourPlot[x^2 + y^2, {x, -5, 5}, {y, -5, 5}]
```



Q2 : Show regions in {x, -1, 1}, {y, -1, 1} of different values of electric potential around a static charge of 5 * 10 ^ -9 C;

Hint :
to solve this question we use the formula of electric potential which related the given data;

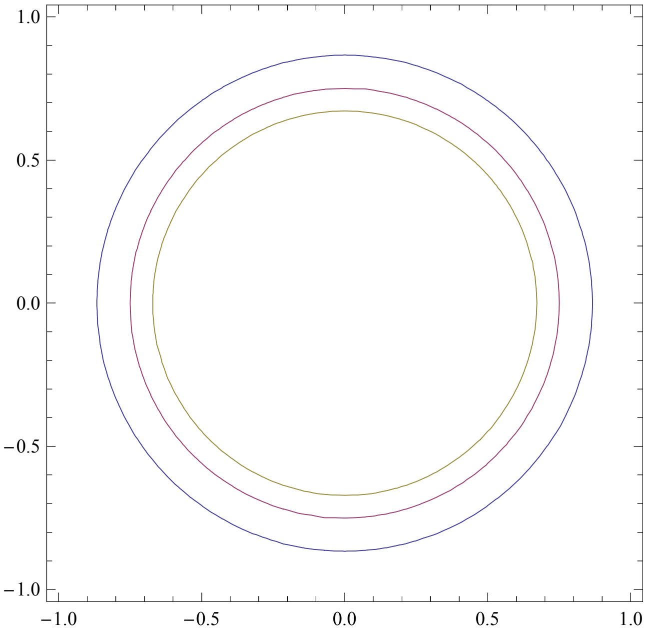
```
k = 9 * 10 ^ 9; q = 5 * 10 ^ -9; ContourPlot[1 / Sqrt[x^2 + y^2] k * q, {x, -1, 1}, {y, -1, 1}]
```



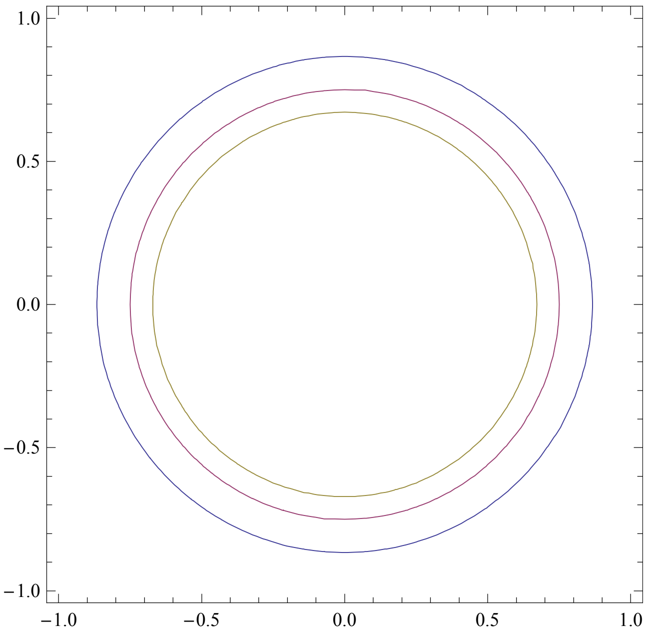
Q3 : draw in {x, -1, 1}, {y, -1, 1} lines of magnitudes of electric field as 60, 80 and 100 N / C around a static charge of 5 * 10 ^ -9 C;

Hint : to solve this question for the different values of electric field we use the third help of ContourPlot commnad;

```
k = 9 * 10 ^ 9; q = 5 * 10 ^ -9;
ContourPlot[{1 / (x^2 + y^2) k * q == 60, 1 / (x^2 + y^2) k * q == 80, 1 / (x^2 + y^2) k * q == 100},
 {x, -1, 1}, {y, -1, 1}]
```

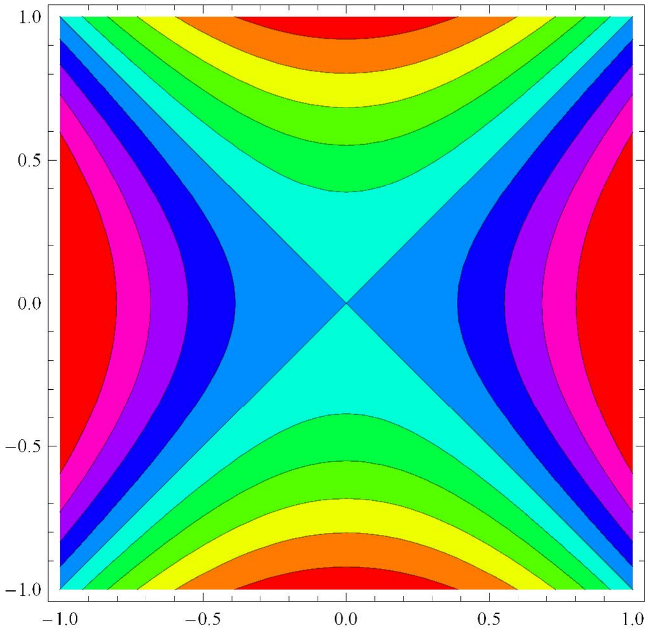


```
f = 1 / (x^2 + y^2) k * q; k = 9 * 10 ^ 9; q = 5 * 10 ^ -9;
ContourPlot[{f == 60, f == 80, f == 100}, {x, -1, 1}, {y, -1, 1}]
```

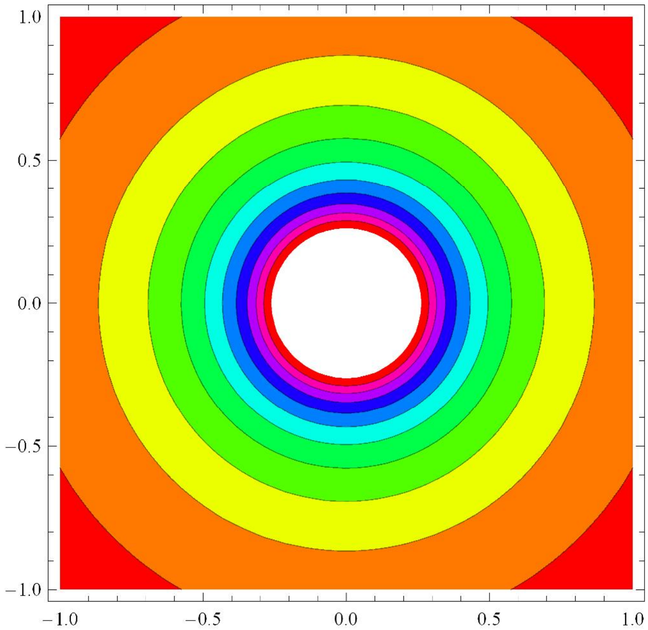


```
HInt : to increase the number of contours ,
colour the contrours and points we use the following typings ;
```

```
ContourPlot[Sin[x^2 - y^2], {x, -1, 1}, {y, -1, 1},
 Contours → 10, ColorFunction → Hue, PlotPoints → 5]
```



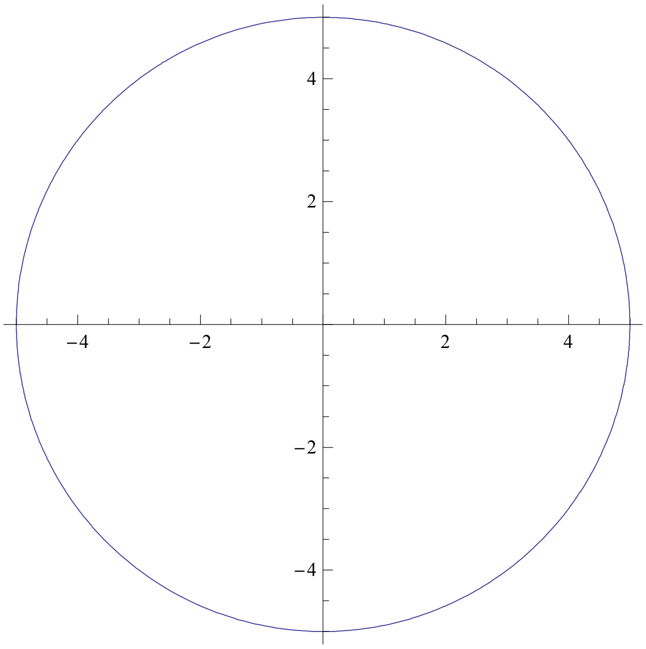

```
k = 9 * 10^9; q = 5 * 10^-9; ContourPlot[1 / Sqrt[x^2 + y^2] k * q,
 {x, -1, 1}, {y, -1, 1}, Contours → 10, ColorFunction → Hue]
```



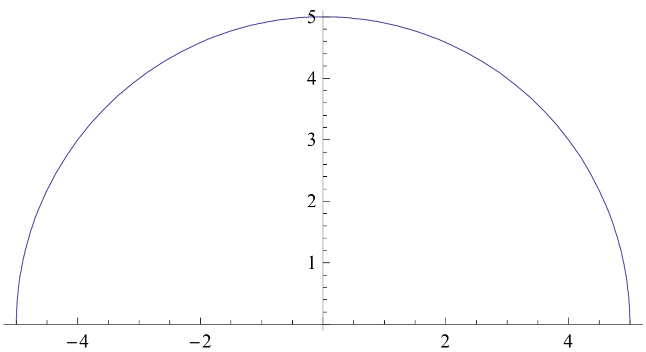
Topic : PolarPlot;

Q : using polar coordinates graph the circle and half of the circle radius of 5 meters;

```
PolarPlot[5, {θ, 0, 2 Pi}]
```

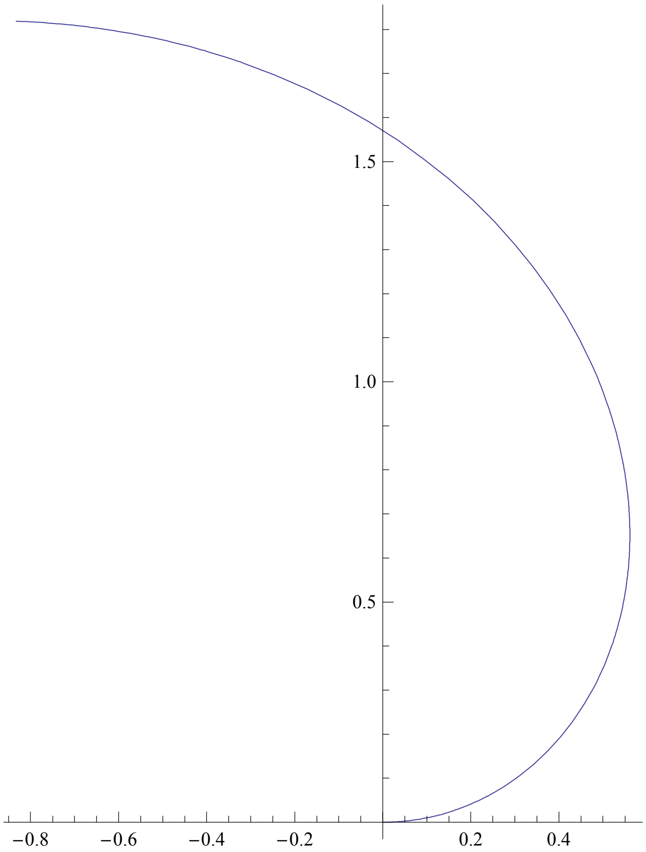



```
PolarPlot[5, {thi, 0, Pi}]
```




```
Topic : LogPlot;
```

```
PolarPlot[x, {x, 0, 2}]
```




```
Topic : grphic data;

Note :  to show the data like points we sue the commnad ListPlot[];
to join the points we use the commnad ListLinePlot[];

?ListPlot
```

ListPlot[{y_1, y_2, …}] plots points corresponding to a list of values, assumed to correspond to x coordinates 1, 2, ….
ListPlot[{{x_1, y_1}, {x_2, y_2}, …}] plots a list of points with specified x and y coordinates.
ListPlot[{$list_1$, $list_2$, …}] plots several lists of points. ≫

```
?ListLinePlot
```

ListLinePlot[{y_1, y_2, …}] plots a line through a list of values, assumed to correspond to x coordinates 1, 2, ….
ListLinePlot[{{x_1, y_1}, {x_2, y_2}, …}] plots a line through specific x and y positions.
ListLinePlot[{$list_1$, $list_2$, …}] plots several lines. ≫

```
Q1 : graph (0.1, 2.3), (0.7, 3.1), (1.1, 2.7),
and also a set of line joining them. display both graphs together;
```

```
g1 = ListPlot[{{0.1, 2.3}, {0.7, 3.1}, {1.1, 2.7}}];
g2 = ListLinePlot[{{0.1, 2.3}, {0.7, 3.1}, {1.1, 2.7}}]; Show[g1, g2]
```