NAME:- NAVEEN R

USN:- 1RV17BT028

QUIZ 3

1. **Calculate the volume of the DNA molecule (Hint: volume of a cylinder)**
   Ans V = pi*((d/2)^2)*h

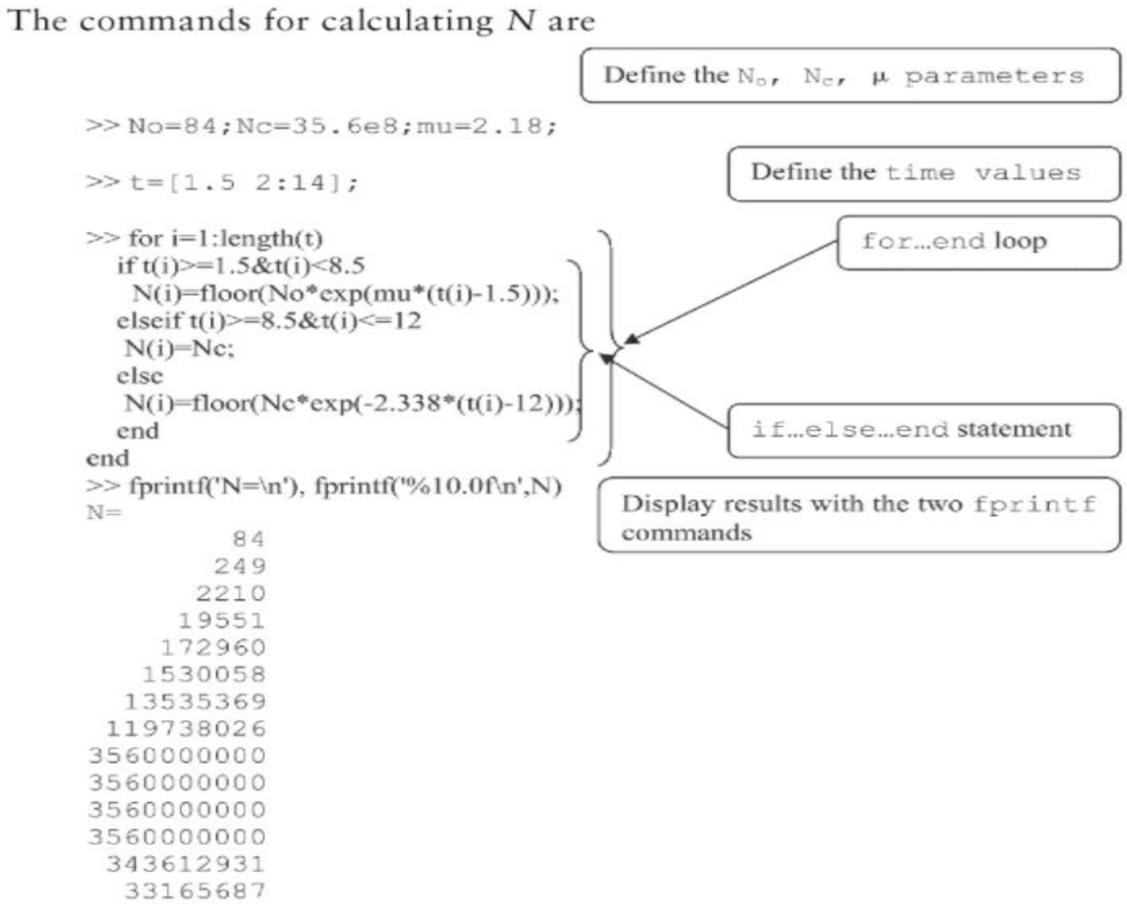
2. **DNA has four bases, adenine, cytosine, guanine and thymine. Problem: Generate and display the matrix in which the first column is the serial number and the second is the base name.**

```
  Ans    serialnumber =[1;2;3;4]

           serialnumber =

        1

        2

        3

        4

 basename = {'Adenine'; 'Cytosine' ; 'Guanine' ; 'Thymine'}

basename =

  4×1 cell array

    {'Adenine' }

    {'Cytosine'}

    {'Guanine' }

    {'Thymine' }


t1 = table(serialnumber, basename)

t1 =

  4×2 table

    serialnumber     basename

    ____________    __________


       1                   'Adenine'
```

2	'Cytosine'
3	'Guanine'
4	'Thymine'

3. **Calculate bacteria populations at 1.5, 2, 3, … , 14 hours. The required steps are:**
   • Determine the variables N0, Nc and μ, and a 1 × 14 vector with the t values;
   • Use the for… end loop in which every pass runs for the new t value defined by its index (address) – t(i);
   • Introduce in the loop the statement if … elseif … else … end in which the conditions and expressions on the right-hand side of the equation defining N should be written in the blank spaces; the values of N should be indexed to generate the vector of calculated N-values for each t;
   • Display with the fprintf the vector of calculated N-values.

Ans:


The commands for calculating *N* are

```
>> No=84;Nc=35.6e8;mu=2.18;

>> t=[1.5 2:14];

>> for i=1:length(t)
   if t(i)>=1.5&t(i)<8.5
     N(i)=floor(No*exp(mu*(t(i)-1.5)));
   elseif t(i)>=8.5&t(i)<=12
    N(i)=Nc;
   else
    N(i)=floor(Nc*exp(-2.338*(t(i)-12)));
   end
end
>> fprintf('N=\n'), fprintf('%10.0f\n',N)
N=
        84
       249
      2210
     19551
    172960
   1530058
  13535369
 119738026
3560000000
3560000000
3560000000
3560000000
 343612931
  33165687
```

Define the No, Nc, μ parameters
Define the time values
for…end loop
if…else…end statement
Display results with the two fprintf commands

4. **The function to find the R.M.S value is _________**

Ans: y = **rms**(x) returns the **root-mean-square** (**RMS**) level of the input, x .

5. **List out the operators that MatLab allows with examples?**

❖ **Arithmetic operator**

- Matrix arithmetic operations
- Array arithmetic operations

Matrix arithmetic operations are same as defined in linear algebra. Array operations are executed element by element, both on one dimensional and multi-dimensional array.

The matrix operators and arrays operators are differentiated by the period (.) symbol. However, as the addition and subtraction operation is same for matrices and arrays, the operator is same for both cases.

➢ +

Addition or unary plus. A+B adds the values stored in variables A and B. A and B must have the same size, unless one is a scalar. A scalar can be added to a matrix of any size.

```
Eg : a = 10;
        b =  20;
        c = a + b;
after running the code output will look like
         c = 30
```

➢ -

Subtraction or unary minus. A-B subtracts the value of B from A. A and B must have the same size, unless one is a scalar. A scalar can be subtracted from a matrix of any size.

```
Eg: a = 10;
       b =  20;
       d = a - b;
after running the code output will look like
         d = -10
```

➢ *

Matrix multiplication. C = A*B is the linear algebraic product of the matrices A and B. More precisely,

$$C(i,j) = \sum_{k=1}^{n} A(i,k)B(k,j)$$

For non-scalar A and B, the number of columns of A must be equal to the number of rows of B. A scalar can multiply a matrix of any size.

```
Eg: a = 10;
       b =  20;
       e = a * b;
after running the code output will look like
         e = 200
```

- **.***

Array multiplication. A.*B is the element-by-element product of the arrays A and B. A and B must have the same size, unless one of them is a scalar.

- /

Slash or matrix right division. B/A is roughly the same as B*inv(A). More precisely, B/A = (A'\B')'.

```
Eg: a = 10;
    b =  20;
    f = a / b;
after running the code output will look like
     f = 0.50000
```

- **./**

Array right division. A./B is the matrix with elements A(i,j)/B(i,j). A and B must have the same size, unless one of them is a scalar.

- \

Backslash or matrix left division. If A is a square matrix, A\B is roughly the same as inv(A)*B, except it is computed in a different way. If A is an n-by-n matrix and B is a column vector with n components, or a matrix with several such columns, then X = A\B is the solution to the equation $AX = B$. A warning message is displayed if A is badly scaled or nearly singular.

```
Eg: a = 10;
    b =  20;
    g = a \ b;
after running the code output will look like
     g = 2
```

- **.**

Array left division. A.\B is the matrix with elements B(i,j)/A(i,j). A and B must have the same size, unless one of them is a scalar.

- ^

Matrix power. X^p is X to the power p, if p is a scalar. If p is an integer, the power is computed by repeated squaring. If the integer is negative, X is inverted first. For other values of p, the calculation involves eigenvalues and eigenvectors, such that if [V,D] = eig(X), then X^p = V*D.^p/V.

```
Eg: x = 7;
    y = 3;
    z = x ^ y;
after running the code output will look like
     z = 343
```

- **.^**

Array power. A.^B is the matrix with elements A(i,j) to the B(i,j) power. A and B must have the same size, unless one of them is a scalar.

➢ '

Matrix transpose. A' is the linear algebraic transpose of A. For complex matrices, this is the complex conjugate transpose.

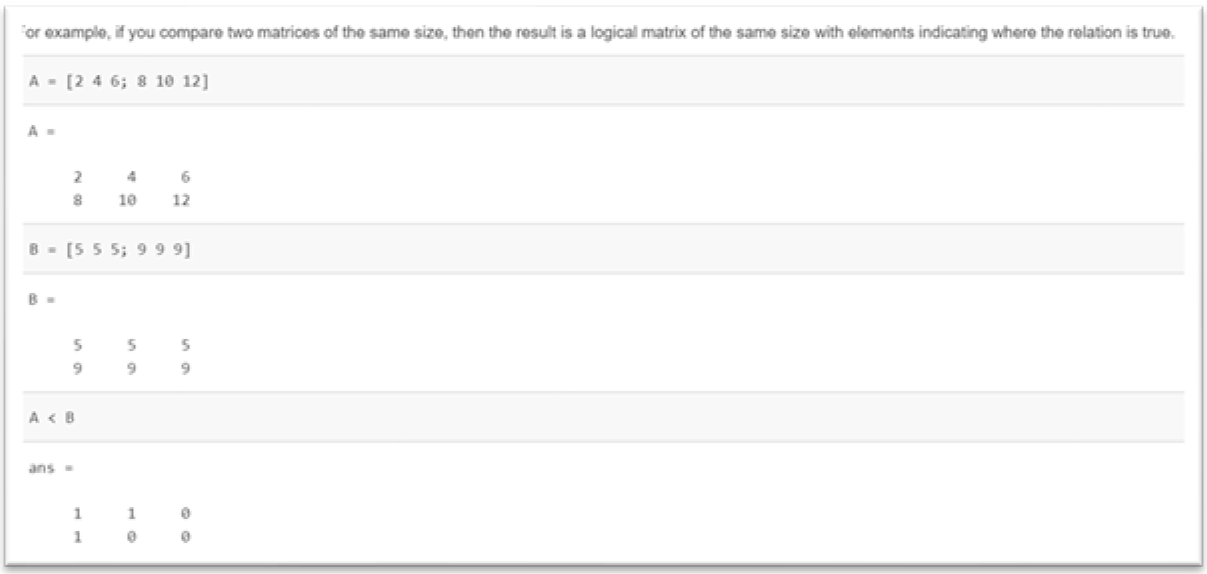
➢ .'

Array transpose. A.' is the array transpose of A. For complex matrices, this does not involve conjugation.

❖ **Relational operator**

These types of Operators can work with both scalar and non-scalar data. As the name suggests it finds a relation between each element of two arrays and if a relation exists then it returns true or else false. The operator returns an array of the same size with values true and false depending on the result of an operation.

- < Less than
- <= Less than equal to
- > Greater than
- >= Greater than or equal to
- == Equal to
- ~= Not equal to



or example, if you compare two matrices of the same size, then the result is a logical matrix of the same size with elements indicating where the relation is true.

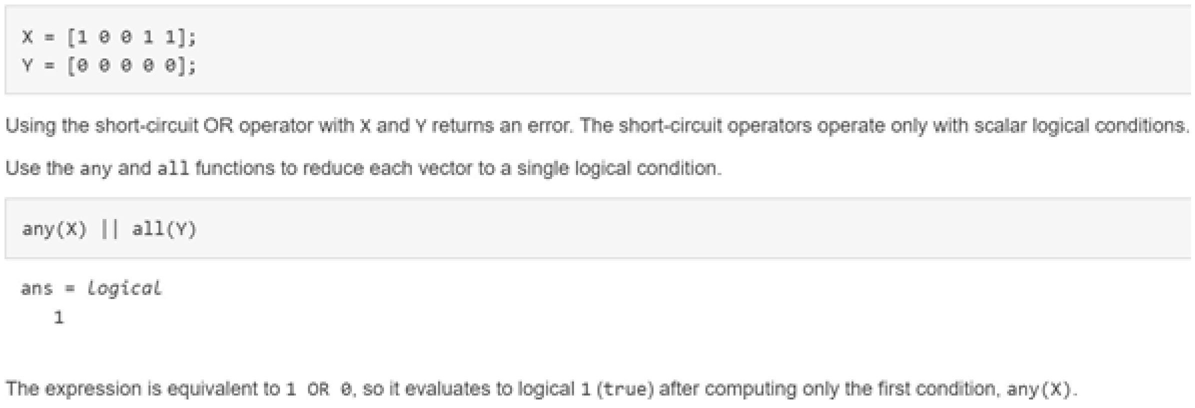
```
A = [2 4 6; 8 10 12]

A =

     2     4     6
     8    10    12

B = [5 5 5; 9 9 9]

B =

     5     5     5
     9     9     9

A < B

ans =

     1     1     0
     1     0     0
```

❖ Logical Operators

Matlab provides two types of Logical Operators are as given below:

➢ **Element-wise:** Element-wise operator operates on elements of logical arrays. The symbols used in these operators are: & (AND), |(OR) ~ (NOT)

➢ **Short-circuit:** These types of operators work on scalar and logical operations. The symbols && and || are the logical short circuit operators AND and OR.



```
X = [1 0 0 1 1];
Y = [0 0 0 0 0];
```

Using the short-circuit OR operator with X and Y returns an error. The short-circuit operators operate only with scalar logical conditions.

Use the any and all functions to reduce each vector to a single logical condition.

```
any(X) || all(Y)
```

```
ans = logical
   1
```

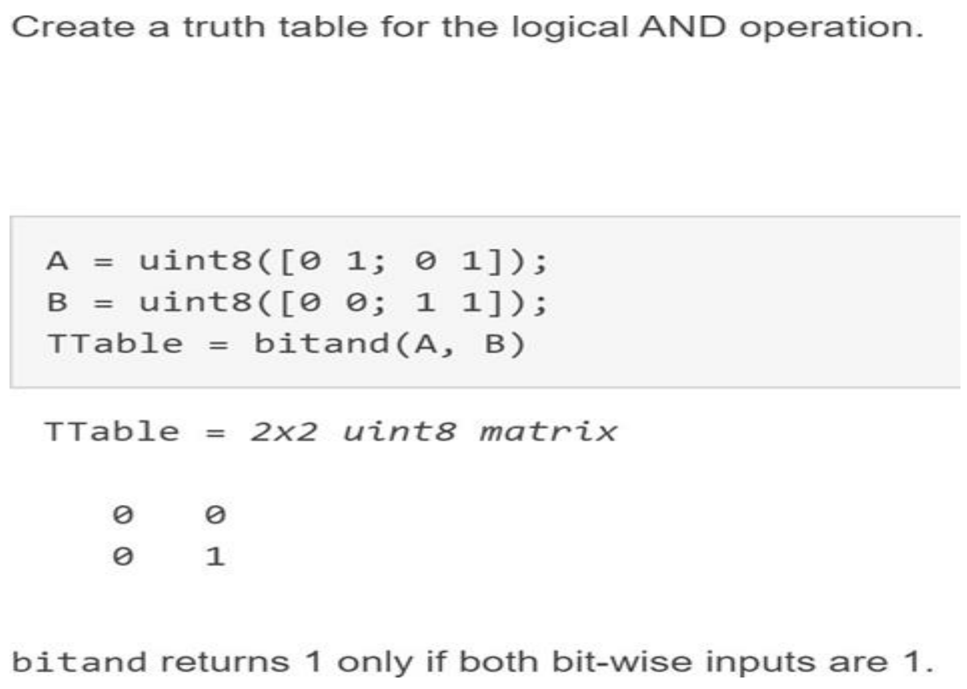
The expression is equivalent to 1 OR 0, so it evaluates to logical 1 (true) after computing only the first condition, any(X).

❖ Bitwise Operators

As it is clear by the name Bitwise Operators work on a bit-by-bit operation. The Bitwise Operator symbols are |, &, and ^:

Interestingly, MATLAB provides various functions for bitwise and, bitwise or, bitwise not operations and shift operation, etc. Matlab provides the following bitwise operators:

- **bit and(a,b)** – Bitwise AND of integers a and b
- **bitmap(a)** – Bitwise complement of a
- **bitget(a, pos)** – Get bit at a specified position, in the array a
- **bitset(a, pos)** – set bit at a specified location of a
- **bitShift(a,k)** – It is equivalent to get multiplied by 2k.So, if k is negative then it shifts right and if k is positive then it shifts left.
- **bitor(a, b)** – Bit-wise XOR of integers a and b



Create a truth table for the logical AND operation.

```
A = uint8([0 1; 0 1]);
B = uint8([0 0; 1 1]);
TTable = bitand(A, B)
```

```
TTable = 2x2 uint8 matrix

    0    0
    0    1
```

bitand returns 1 only if both bit-wise inputs are 1.

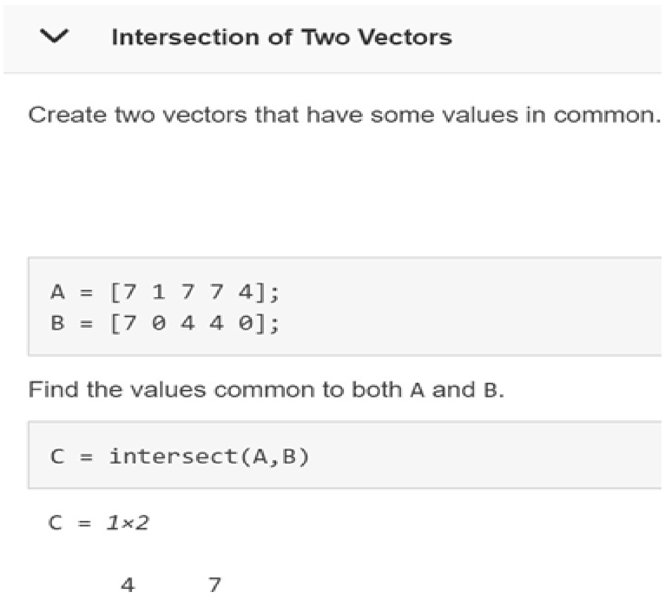
- **Set Operators**

MATLAB provides various set operators like a union, intersection, etc. Please

find below the various list of operations:

- **intersect(A & B):** It is used to intersect A and B and returns the common

  values of A and B in sorted order.
- **intersect(A & B, ‘rows’):** It returns rows common to both A and B.
- **ismember(A, B):** It returns an array of size A with 1 for all elements of A that are found in B or else none.
- **Issorted(A):** It returns 1 if elements are in sorted order and 0 if not.
- **union**: It sets union of two array
- **unique:** Gives unique values in an array.



Intersection of Two Vectors

Create two vectors that have some values in common.

```
A = [7 1 7 7 4];
B = [7 0 4 4 0];
```

Find the values common to both A and B.

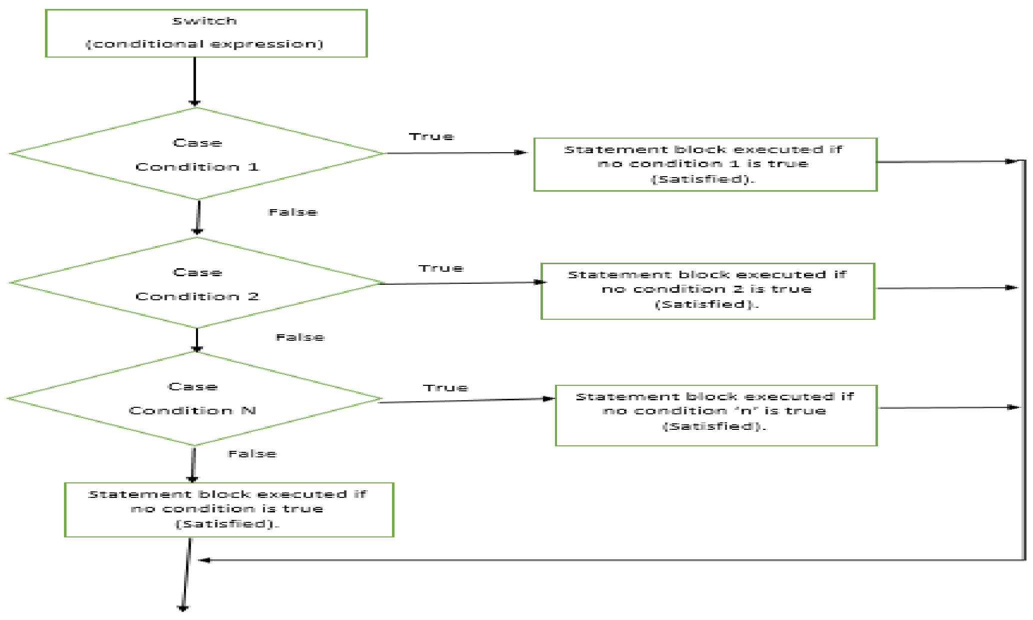
```
C = intersect(A,B)
```

```
C = 1×2

     4     7
```

6. Write Xmath features

Ans: Following are the Xmath features:
1. Sripting languages with OOP features.
2. Libraries that are LNX and C language compatible.
3. A debugging tools with GUI features.
4. Color graphics can be pointed and clickable.
5. A special layer is available that is programmable for MOTIF GUI.

7. Draw flow diagram for Switch statement in Matlab and give a example for switch Statement


Switch
(conditional expression)
Case
Condition 1
True
Statement block executed if no condition 1 is true (Satisfied).
False
Case
Condition 2
True
Statement block executed if no condition 2 is true (Satisfied).
False
Case
Condition N
True
Statement block executed if no condition 'n' is true (Satisfied).
False
Statement block executed if no condition is true (Satisfied).

Example:

```
N = input('Enter a number of your choice: ');
switch N
case -2
disp('negative one selected')
case 0
disp('zero selected')
case 2
disp('positive one selected')
otherwise
disp('Some other value')
end
```

Output:

At the command prompt, enter the number -2.

```
negative two
```

Repeat the code and enter the number 5.

```
Some other value
```