iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 7 **Issue:** XII **Month of publication:** December 2019

DOI: http://doi.org/10.22214/ijraset.2019.12046

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.177
Volume 7 Issue XII, Dec 2019- Available at www.ijraset.com

DNA-FAM: Your Search to the World of Dogma

Subhojyoti Chatterjee[1], Jagriti Chatterjee[2]
[1]*School of Biotechnology, Amrita Vishwa Vidhyapeetham, Amritapuri Campus, Clappana P.O. Kollam – 690 525, Kerala, India.*
[2]*Department of Computational Biology and Bioinformatics, Sam Higginbottom University of Agriculture, Technology and Sciences, Rewa Road, Naini, Allahabad-211007, Uttar Pradesh, India.*

Abstract: ***The building blocks of life such as deoxyribonucleic acid (DNA) bases elaborate biological processes with the living cells. Here we have attempted to describe the gene mutation and six possible open reading frames (ORF) of DNA using the platform of PERL programming. The programming package DNA-FAM will not only help in finding better similarity between the DNA sequences but will also shed light in comprehending the phenotypic and genotypic alterations arising from random mutations.***

Keywords: Deoxyribonucleic acid (DNA), Ribonucleic acid (RNA), Open reading frame (ORF), Mutation, PERL programming.

I. INTRODUCTION

Nucleic acids are important to living cells. The alluring question from a chemical standpoint has been the heredity study involving the conveyance of information from one generation to the adjoining one, through the uncovering of heredity material "chromosomes" that dates back to the 19th century. Being present in the nucleus of a living cell which is acidic in nature, they are primarily composed of two types of nucleic acid – deoxyribonucleic acid (DNA) and ribonucleic acid (RNA), differing in function and chemical composition of the sugar moiety being 2-deoxyribose and ribose, respectively. Nucleotides, the nucleic acid subunits, are linear polymeric molecules of three polynucleotide subunits (Fig. 1) composed of a nitrogenous base (a heterocyclic aromatic compound containing nitrogen), a 5-carbon pentose sugar and a 3-phosphate group (a phosphoric acid derivative). The phosphodiester bonds (formed between nucleotides) i.e. the linkage between the 5′, 3′ carbons of the ribose ring are responsible for the construction of ATP (adenosine triphosphate), the energy unit of the cell. Such sequences are called the primary structure of biopolymers [1].

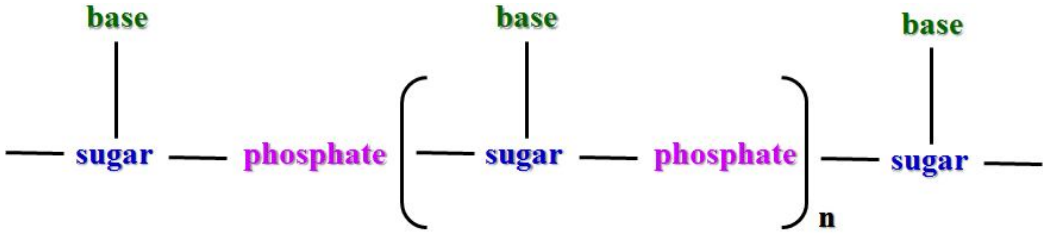


Fig. 1 Schematic representation of a nucleic acid chain

The DNA/RNA bases such as, adenine (A), guanine (G), cytosine (C), thymine (T) and uracil (U) are the different types of nucleotides that are phosphorylated by specific enzymes in the cell (Fig. 2(a)). Of these, the DNA double helix structure (Fig. 2(a)) by Watson-Crick (1953) [2] is dictated by base-pairing of A with T and C with G, known as B-form DNA. Interestingly, cytosine (C), thymine (T) and uracil (U for RNA, discussed below) share a pyrimidine pentagon ring fusion [1], making pyrimidine an important core component for the DNA bases [4-7]. The double stranded secondary helix structure of DNA [8-11] resembles a flexible ladder with two nucleic acid chains wound about each other and interconnected through hydrogen bonds between 2-deoxyribose bases, as decoded by Francis Crick [2] and James Watson [2] in 1953, for which they were honored with a Nobel Prize in 1962. In contrast, RNA forms complexly large structures of proteins including loose single strands of messenger RNA (m-RNA) with locally folded regions, where the fourth base is U rather than T (Fig. 2(a)) [12]. They also have different contrasting biological functional RNA, i.e., transfer RNA (*t*-RNA) and ribosomal RNA (*r*-RNA), which are capable of performing enzymatic catalysis and was disclosed by Tom Cech (1982) [13], initially describing them as "ribozymes". These complex structures are facilitated by significant distinctive conformations of RNA backbone which are less locally flexible, due to the extra OH on the ribose along with both (+) / (-) interactions [14].

The biological function of nucleic acids mainly involves (i) DNA replication (Fig. 2(b)) and (ii) protein synthesis, wherein there is a self-duplication of a DNA molecule during cellular division. The replication process [15-17] is started by the parent DNA, unwinding the two strands to serve as a master template (called "replisomes") for new partner construction from 5′ to 3′ direction

©IJRASET: All Rights are Reserved

known as the replication fork. Since there is a specific nucleic acid base pairing of A=T/G≡C, the newly built strands are complimentary to the old ones and thus contribute to the double helical structure [8] with one parent old strand along with the new one, being transmitted to the daughter cell (Fig. 2(b)). Protein synthesis [18-20], another crucial function of nucleic acids involves generation of new proteins through balanced degradation of bio-cellular ones by blending coded information from specific DNA base sequences. In this process, there is the transmission of information from DNA to messenger RNA (*m*-RNA), which goes to the cell cytoplasm through the essential biosynthetic pathway. Messenger RNA (*m*-RNA) acts as a template for proper protein sequences by bringing the amino acids inside the cell ("translation") by transfer RNA (*t*-RNA) [18] where they form peptide bonds. Thus, in a nutshell, it is the DNA that contains the coded message for protein synthesis, whereas RNA performs the carrying out of the synthesis process.

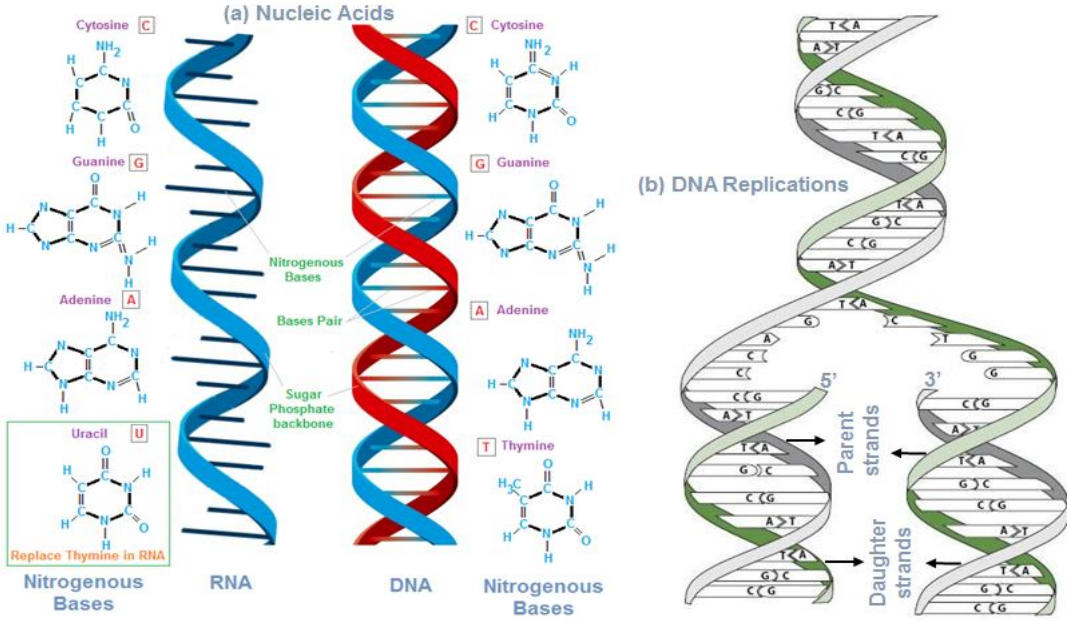


Fig. 2 (a) Nucleic acids i.e. Watson and Crick's double helix structure DNA along with single-stranded RNA and (b) Replication of DNA.

The replication process in DNA sometimes leads to misfitting of conjugate bases, leading to mutation [17]. Mutation is a change which leads to both phenotypic and genotypic alterations, mechanism by which it occurs is random. Mutation is a change which could be spontaneous or induced leading to change in the reading frame. DNA can be read in six frames i.e., (+3, +2, +1 and -3, -2, -1). This helps in finding better similarity and identity among sequences. Performing sequence similarity match is an important step towards finding the homology among sequences. The aim of this present work is to develop a code for performing different functionalities over double helix structure DNA, using the platform of programming language PERL. The programming depicts how randomly the DNA gets mutated with the bases. Multiple sequence alignment (MSA) and local alignment (LA) are done to find the percentage similarity and identity between the sequences. It is about finding the percentage with respect to the position that are similar between these DNA sequences. Therefore, the programming package (DNA-FAM) shows: (a) how mutation occurs randomly in DNA (a gene mutation), (b) demonstrating six possible open reading frames of DNA and (c) percentage identity between random DNA sequences.

II. COMPUTER SPECIFICATIONS TO RUN DNA-FAM

To run the programming package DNA-FAM, the following minimum computer specification is required, i.e.,

1) Operating System (OS) – Windows 10
2) System Type – 64-bit OS, x64-based processor
3) Random Access Memory (RAM) – 8.00 GB
4) Processor – Intel® Core™ i5-6200U CPU @ 2.30GHz
2.40 GHz
5) Perl Compiler for Windows – Strawberry Perl and ActiveState Perl (https://www.perl.org/get.html)

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.177
Volume 7 Issue XII, Dec 2019- Available at www.ijraset.com

III.DNA-FAM PROGRAMMING MODULE

There are four different programming modules present within the main programming domain of DNA-FAM. The first module *MUTATE.PM* performs the process of random mutation upon the DNA sequences during the process of replication, the second module *RANDOM.PM* generates DNA sequences upon which phenotypic and genotypic alterations are studied, the third module *SIMILARITY.PM* calculates the percentage similarity and identity of the DNA sequences based on multiple sequence alignment and local alignment algorithms and finally the fourth module SIXFRAME.PM translates the codon on a strand of DNA that specifies the genetic code using open reading frame. These four modules are incorporated in the main programming domain of DNA-FAM through which the subroutines are called, and the DNA functionalities are been executed. The details of the source code for the main programming module and the four subroutines are illustrated below.

A. Source Code for Main Programming Domain DNA-FAM

```
use mutate;
use random;
use similarity;
use sixframe;

#!usr/bin/perl

do

{
        print "\n=================================================================\n";

 print "    ****** WELCOME ****** TO ****** DNA FAM ****** WORLD ******\n";

print"=================================================================\n\n";
print "   *!*!*!*!*  OPTIONS FOR UR DNA FAM FUNCTIONALITIES  *!*!*!*!*\n\n";
        print "!!!
```

```
	my $mutant;

	# seed the random number generator
	# time|$$ combines the current time with the current process id
	srand(time|$$);

	# lets test it,shall we?
	$mutant = mutate($dna1);
	print "\n\n\t\t ----------------------------------------------------";
	print "\n\t\t <===== MUTATTION PROCESS =====>\n";
	print "\t\t ----------------------------------------------------\n\n";
	print "\nThe Original DNA ===>\n\n";
	print "$dna1\n";

	print "\nThe Mutant DNA ===>\n\n";
	print "$mutant\n";

	# let's put it in a loop and watch that bad boy accumulate mutations
	print "\n~~~
```

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.177
Volume 7 Issue XII, Dec 2019- Available at www.ijraset.com

```
srand(time|$$);

# and here's the subroutine call to do the real work
@random_DNA = make_random_DNA_set( $minimum_length,
                                   $maximum_length, $size_of_set );

# print the results, one per line
print"\n**********************************************************************\n\n";
print "\t** Here Is An Array Of ** $size_of_set ** Randomly Generated DNA
       Sequences **\n\n";
print "\t\t*** with lengths between *** $minimum_length *** and ***
      $maximum_length *** \n\n";
print"**********************************************************************\n\n";
foreach my $dna(@random_DNA)
{
          print "$dna\n\n";
}


# to look for % similarity in DNA

# calculate the average percentage of positions that are the same
# between two random DNA sequence,in a set of 10 sequence

# declare and initialize the variables
my $percent;
my @percentages;
my $result;


# iterate through all pairs of sequence
for (my $k = 0 ; $k < scalar @random_DNA - 1 ; ++$k)
{
          for (my $i = ($k + 1); $i < scalar @random_DNA ; ++$i)
          {

                    # calculate and save the matching percentage
                    $percent = matching_percentage($random_DNA[$k],
                                                   $random_DNA[$i]);
                    push(@percentages, $percent);
          }
}

# finally, the average result
$result = 0;


foreach $percent(@percentages)
{
          $result += $percent;
```

©IJRASET: All Rights are Reserved

```
	}

	$result = $result/scalar(@percentages);
	# turn result into a true percentage
	$result = int($result * 100);

	print "\tIn This Run Of The Experiment,The Average Percentage Of \n\n";
	print "\t\tMatching Positions Is =====>  $result%\n\n";
	print"<><><>
```

```
$r4=$rev;
print "\t~~~~~~~~~~~~~~~~ READING FRAME 4 ~~~~~~~~~~~~~~~~\n";
print $r4;
print "\n\n";

shift(@rev);
$r5=join(undef,@rev);
print "\t~~~~~~~~~~~~~~~~ READING FRAME 5 ~~~~~~~~~~~~~~~~\n";
print $r5;
print "\n\n";

shift(@rev);
$r6=join(undef,@rev);
print "\t~~~~~~~~~~~~~~~~ READING FRAME 6 ~~~~~~~~~~~~~~~~\n";
print $r6;
print "\n\n";



@trans=("$r1","$r2","$r3","$r4","$r5","$r6");
$j=1;
foreach $val(@trans)
{
        $l=length($val);
        print"len=$l\n\n";
        $i=0;
        while($i<$l)
        {

                $str=substr($val,$i,3);
                $i=$i+3;
                print "str: $str\n\n";
                if(length($str)==3)
                {
                        $amino=&translate_codon($str);
                        push(@prot,$amino);
                }

        }

        print "READING FRAME $j :\n\n";
        $j=$j+1;
        $len=@prot;
        print "length:$len\n\n";
        $prot=join(undef,@prot);
        $prot=~s/\s*//g;
        print "$prot";
        @prot=();
}
print"\n\n";
```

©IJRASET: All Rights are Reserved

```
        }

        else

        {
                print "\t\t Sorry!!! plz.... Enter A Correct Choice \n\n"
        }
}while($choice ne 4);

print "\n************* THANK YOU FOR USING OUR SERVICE ************\n\n";
print"==========================================================\n\n";
```

B. Source Code for First Subroutine MUTATE.PM

The subroutine first selects a random position in a string of DNA using random position subroutine and calls the DNA strand to seed the random number generator. This helps in the initiation of point mutation. It is followed by selection of a random nucleotide which is placed into a random position using the mutate subroutine, to unfold the process of mutation and disease abnormalities attached with it, during DNA replication.

```
package mutate;
require Exporter;
@ISA=qw(Exporter);
@EXPORT=qw(mutate randomelement randomnucleotide randomposition);

BEGIN
{
        push(@INC,"/home/perl_DNAFAM_project");
}

##############################################################################
#                subroutines begin for mutant dna                            #
##############################################################################


# a subroutine to perform a mutation in a string of dna
#
# WARNING: make sure you call srand to see the
# random number generator before you call this function

sub mutate {

        my($dna) = @_;

        my(@nucleotides) = ('A', 'C', 'G', 'T');

        # pick a random position in the dna
        my($position) = randomposition($dna);

        # pick a random nucleotide
        my($newbase) = randomnucleotide(@nucleotides);

        # insert the random nucleotide into the random position in the dna
```

©IJRASET: All Rights are Reserved

```
        # the substr arguments mean the following
        # in the string $dna at position $position change 1 character to
        # the string in $newbase
        substr($dna,$position,1,$newbase);

        return $dna;
}
# a subroutine to randomly select an element from an array
#
# WARNING: make sure you call srand to seed the
# random number generator before you call yhis function

sub randomelement {

        my(@array) = @_;

        return $array[rand @array];
}

# randomnucleotide
#
# a subroutine to select at random one of the four nucleotides
#
# WARNING: make sure you call srand to seed the
# random number generator before you call this function

sub randomnucleotide {

        my(@nucleotides) = ('A', 'C', 'G', 'T');

        # scalar returns the size of an array
        # the elements of the array are numbered 0 to size-1
        return randomelement(@nucleotides);
}

# randomposition
#
# a subroutine to randomly select a position in a string
#
# WARNING: make sure you call srand to seed the
# random number generator before you call this function

sub randomposition {

        my($string) = @_;

        # notice the "nested" arguments
        #
        # $string is the argument to length
        # length($string) is the argument to rand
```

©IJRASET: All Rights are Reserved

```
        # rand(length($string))) is the argument to int
        # int(rand(length($string))) is the argument to return
        # but we write it without parentheses, as permited
        #
        # rand returns a decimal number between 0 and its argument
        # int returns the integer portion of a decimal number
        #
        # the whole expression returns a random number between 0 and length-1
        # which is howposition in a string are numbered in perl
        #
        return int rand length $string;
}
```

C. Source Code for Second Subroutine RANDOM.PM

An array is initialized to store the DNA sequence. The random number generator is seeded (a) to make a set of random DNA, (b) to accept parameters setting the maximum and minimum length of each string of DNA, and (c) to select one nucleotide / element randomly from the array, upon which phenotypic and genotypic alterations are studied.

```
package random;
require Exporter;
@ISA=qw(Exporter);
@EXPORT=qw(make_random_DNA_set randomlength make_random_DNA randomnucleotide randomelement);

BEGIN
{
        push(@INC,"/home/perl_DNAFAM_project");
}

##############################################################################
#                    subroutines for generation of random dna                #
##############################################################################

# make_random_DNA_set
#
# make a set of random DNA
#
# accept parameters setting maximum and minimum length of
# each strings of DNA, and the number of DNA strings to make
#
# WARNING: make sure you call srand to seed the
# random number generator before you call this function

sub make_random_DNA_set {

        # collect arguments, declare variables
        my($minimum_length, $maximum_length, $size_of_set) = @_;

        # length of each DNA fragment
        my $length;
```

©IJRASET: All Rights are Reserved

```
        # DNA fragments
        my $dna;

        # set of DNA fragments
        my @set;

        # create set of random DNA
        for (my $i = 0; $i < $size_of_set; ++$i) {

        # find a random length between min and max
        $length = randomlength($minimum_length, $maximum_length);

        # make a random DNA fragment
        $dna = make_random_DNA( $length );

        # add $dna fragment to @set
        push( @set, $dna );
        }
return @set;
}

# needed: randomlength, which will return a random
# number between(or including)the min and max values

# randomlength
#
# a subroutine that will pick a random number from
# $minlength to $maxlength, inclusive
#
# WARNING: make sure you call srand to seed the
# random number generator before you call this function

sub randomlength {

        # collect arguments, declare variables
        my($minlength, $maxlength) = @_;

        # calculate and return a random number within the
        # desired interval,
        # notice how we need to add one to make the endpoints inclusive
        # and how we first subtract,then add back, $minlength to
        # get the random number in the correct interval
        return( int(rand($maxlength - $minlength + 1)) + $minlength );
}

# make_random_DNA
#
# make a string of random DNA of specified length
#
# WARNING: make sure you call srand to seed the
```

©IJRASET: All Rights are Reserved

```
# random number generator before you call this function

sub make_random_DNA {

	# collect arguments, declare variables
	my($length) = @_;

	my $dna;

	for (my $i=0 ; $i < $length ; ++$i) {

		$dna.= randomnucleotide();
	}
return $dna;
}

# randomnucleotide
# here it is there for completeness

# randomnucleotide
#
# select at random one of the four nucleotides
#
# WARNING: make sure you call srand to seed the
# random number generator you call this function

sub randomnucleotide {

	my(@nucleotides) = ('A', 'C', 'G', 'T');

	# scalar returns the size of an array
	# the elements of the array are numbered 0 to size-1
	return randomelement(@nucleotides);
}

# randomelement
#
# randomly select an element from an array
#
# WARNING: make sure you call srand to seed the
# random number generator before you call this function

sub randomelement {

	my(@array) = @_;

	return $array[rand @array];
}
```

©IJRASET: All Rights are Reserved

D. Source Code for Third Subroutine SIMILARITY.PM

An array is initialized to store the DNA. Seed the random number generator and generate the data set of sequences along with its iteration through all the pairs. The percentage of matching positions are then calculated. The subroutine matching_percentage is evoked to calculate the percentage of identical bases in two equal lengths of DNA sequences. This is followed by calling subroutine make_random_dna_set, for making a set of random DNAs. Furthermore, the random number is picked by using the subroutine randomlength. Finally, the subroutine make_random_dna is called to make a random DNA of specific length which is then preceded by selecting a random nucleotide from an array by the functionality of subroutine randomnucleotide and randomelement, for calculation of percentage identity between random DNA sequences.

```
package similarity;
require Exporter;
@ISA=qw(Exporter);
@EXPORT=qw(matching_percentage make_random_DNA_set randomlength make_random_DNA randomnucleotide
randomelement);

BEGIN
{
	push(@INC,"/home/perl_DNAFAM_project");
}

##############################################################################
#		subroutine for checking percentage similarity between the sequences		#
##############################################################################

# matching_percentage
#
# subroutine to calculate the percentage of identical bases in two
# equal length DNA sequence

sub matching_percentage {

	my($string1, $string2) = @_;

	# we assume that the strings have the same length
	my($length) = length($string1);
	my($position);
	my($count) = 0;

	for($position=0; $position < $length; ++$position) {
		if(substr($string1,$position,1) eq substr($string2,$position,1)) {
			++$count;
		}
	}
	return $count / $length;
}

# make_random_DNA_set
#
# subroutine to make a set of random DNA
#
# accept parameters setting the maximum and minimum length of
```

©IJRASET: All Rights are Reserved

```
# each string of DNA and the number of DNA strings to make
#
# WARNING: make sure you call srand to seed the
# random number generator before you call this function

sub make_random_DNA_set {

	# collect arguments, declare variables
	my($minimum_length, $maximum_length, $size_of_set) = @_;

	# length of each DNA fragment
	my $length;

	# DNA fragment
	my $dna;

	# set of DNA fragments
	my @set;

	# create set of random DNA
	for (my $i = 0; $i < $size_of_set; ++$i) {

		# find a random length between min and max
		$length = randomlength($minimum_length, $maximum_length);

		# make a random DNA fragment
		$dna = make_random_DNA( $length );

		# add $dna fragment to @set
		push( @set, $dna );
	}

	return @set;
}

# randomlength
#
# a subroutine that will pick a random number from
# $minlength to $maxlength inclusive.
#
# WARNING: make sure you call srand to seed the
# random number generator before you call this function

sub randomlength {

	# collect arguments, declare variables
	my($minlength, $maxlength) = @_;

	# calculate and return a random numberwithin the
	# desired interval
```

©IJRASET: All Rights are Reserved

```
        # notice how we need to add one to make the endpoints inclusive,
        # and how we first subtract, then add back, $minlength to
        # get the random number in the correct interval
        return( int(rand($maxlength - $minlength + 1)) + $minlength );
}
# make_random_DNA
#
# make a string of random DNA of specified length
#
# WARNING: make sure you call srand to seed the
# random number generator before you call this function

sub make_random_DNA {

        # collect arguments, declare variables
        my($length) = @_;

        my $dna;
        for (my $i=0 ; $i < $length ; ++$i) {

                $dna.= randomnucleotide();
        }
return $dna;
}
# randomnucleotide
#
# select at random one of the four nucleotides
#
# WARNING: make sure you call srand to seed the
# random number generator you call this function

sub randomnucleotide {
        my(@nucleotides) = ('A', 'C', 'G', 'T');

        # scalar returns the size of an array
        # the elements of the array are numbered 0 to size-1
        return randomelement(@nucleotides);
}
# randomelement
#
# randomly select an element from an array
#
# WARNING: make sure you call srand to seed the
# random number generator before you call this function

sub randomelement {
        my(@array) = @_;
        return $array[rand @array];
}
```

©IJRASET: All Rights are Reserved

E. Source Code for Fourth Subroutine SIXFRAME.PM

A portion of DNA which contains no stop codon (when translated into amino acids) is known as an open reading frame (ORF). The DNA sequences genetic codes are read in group of three base pairs, i.e., the double-stranded DNA can read in any of the six possible open reading frames. It reads three in the forward direction and three in the reverse direction. A long ORF is likely to be part of a gene. As the process begins with the translation of DNA into amino acids, therefore the subroutine translate codon is been activated. Furthermore, an array is initialized and validated. Finally, in a loop the subroutine translate codon is called and the open reading frames are achieved.

```
package sixframe;
require Exporter;
@ISA=qw(Exporter);
@EXPORT=qw(translate_codon);

BEGIN
{
        push(@INC,"/home/perl_DNAFAM_project");
}

##############################################################################
#                    subroutine to translate dna to six frame                #
##############################################################################

sub translate_codon {

$codon=shift;
print "codon: $codon \t";
if ( $codon =~m /GC[AGCU]/i ) { return A;} # Alanine;
if ( $codon =~m/UGC|UGU/i ) { return C;} # Cysteine
if ( $codon =~m/GAC|GAU/i ) { return D;} # Aspartic Acid;
if ( $codon =~m/GAA|GAG/i ) { return E;} # Glutamine;
if ( $codon =~ m/UUC|UUU/i ) { return F;} # Phenylalanine;
if ($codon =~m/GG[AGCU]/i ) { return G;} # Glycine;
if ( $codon =~ m/CAC|CAU/i ) { return H;} # Histine (start codon);
if ( $codon =~ m/AU[AUC]/i ) { return I;} # Isoleucine;
if ( $codon =~ m/AAA|AAG/i ) { return K;} # Lysine;
if ( $codon =~ m/UUA|UUG|CU[AGCU]/i ) { return L;} # Leucine;
if ( $codon =~ m/AUG/i ) { return M;} # Methionine;
if ( $codon =~ m/AAC|AAU/i ) { return N;} # Asparagine;
if ( $codon =~ m/CC[AGCU]/i ) { return P;} # Proline;
if ( $codon =~ m/CAA|CAG/i ) { return Q;} # Glutamine;
if ( $codon =~ m/AGA|AGG|CG[AGCU]/i ) { return R;} # Arginine;
if ( $codon =~ m/AGC|AGU|UC[AGCU]/i ) { return S;} # Serine;
if ( $codon =~ m/AC[AGCU]/i ) { return T;} # Threonine;
if ( $codon =~ m/GU[AGCU]/i ) { return V;} # Valine;
if ( $codon =~ m/UGG/i ) { return W;} # Tryptophan;
if ( $codon =~ m/UAC|UAU/i ) { return Y;} # Tyrosine;
if ( $codon =~ m/UAA|UGA|UAG/i ) { return "***" ;} # Stop Codons;

}
```

©IJRASET: All Rights are Reserved

IV. DNA-FAM PROGRAMMING OUTPUT

This program version allows the users to compute the functionalities on DNA sequences by giving us an insightful knowledge about the phenotypic and genotypic alterations arising due to the mutation in DNA. The functionalities have been demonstrated in the DNA-FAM output based on the input choices from the users:

```
======================================================================
   ****** WELCOME ****** TO ****** DNA FAM ****** WORLD ******
======================================================================

  *!*!*!*!*  OPTIONS FOR UR DNA FAM FUNCTIONALITIES  *!*!*!*!*

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
(1). Mutation Of DNA
(2). To Generate Random Sequence & Find Percentage Similarity
(3). To read DNA in Six Frame
(4). Exit
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
plz... enter ur choice ====> 1
======================================================================


<=== You Have Selected The Option (1) To Cause DNA Mutation ===>
Enter the DNA sequence ===>
ATGCTTTCCCCAGGAGTTCAGGGGAAACCT

                    ---------------------------------------------------
                    <===== MUTATTION PROCESS =====>
                    ---------------------------------------------------


The Original DNA ===>
ATGCTTTCCCCAGGAGTTCAGGGGAAACCT

The Mutant DNA ===>

ATGCTTTCCCCAGGAGTTCAAGGGAAACCT
~~~
```

```
======================================================================
   ****** WELCOME ****** TO ****** DNA FAM ****** WORLD ******
======================================================================

  *!*!*!*!*  OPTIONS FOR UR DNA FAM FUNCTIONALITIES  *!*!*!*!*

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
(1). Mutation Of DNA
(2). To Generate Random Sequence & Find Percentage Similarity
(3). To read DNA in Six Frame
(4). Exit
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

plz... enter ur choice ====> 2

======================================================================
<=== You Have Selected The Option (2) To Generate Random DNA ===>
<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>

Enter The No Of DNA Sequence To Be Generated ====>       10


Enter The Maximum Length Of The DNA ====>      15


Enter The Minimum Length Of The DNA ====>      10

**********************************************************************

          ** Here Is An Array Of ** 10 ** Randomly Generated DNA Sequences **

                    *** with lengths between *** 10 *** and *** 15 ***

**********************************************************************

TGACTTGTGCTTATA

AGATTGTGCGCA

TCCGAGGCGTAGTA

CGAACCCATCAAGC

ATCAGCCGAAGCC

CTCGAGCGAAGATTT

CGTAGTGGAAC

TGCACGACCTA
```

©IJRASET: All Rights are Reserved

TGCTACCGTTG

CGTTTTTAGCACG

In This Run Of The Experiment,The Average Percentage Of

Matching Positions Is =====> 22%

<><><>

GCCUTCUGUCGUGCGGCGGCTCCUTCCCGTGGTGUGGGTCCCTCUUCCCCGUCCTGTGCCTCUTCGCGCGGUCGTUU
CuutctCCuug

~~~~~~~~~~~~~~~ READING FRAME 5 ~~~~~~~~~~~~~~~~~

CCUTCUGUCGUGCGGCGGCTCCUTCCCGTGGTGUGGGTCCCTCUUCCCCGUCCTGTGCCTCUTCGCGCGGUCGTUUCu
utctCCuug

~~~~~~~~~~~~~~~ READING FRAME 6 ~~~~~~~~~~~~~~~~~

CUTCUGUCGUGCGGCGGCTCCUTCCCGTGGTGUGGGTCCCTCUUCCCCGUCCTGTGCCTCUTCGCGCGGUCGTUUCuu
tctCCuug

len=88

str: guu

codon: guu str: CCt

codon: CCt str: ctu

codon: ctu str: uCU

codon: uCU str: UTG

codon: UTG str: CUG

codon: CUG str: GCG

codon: GCG str: CGC

codon: CGC str: TUC

codon: TUC str: TCC

codon: TCC str: GTG

codon: GTG str: TCC

codon: TCC str: UGC

codon: UGC str: CCC

codon: CCC str: UUC

codon: UUC str: TCC

codon: TCC str: CTG

codon: CTG str: GGU

codon: GGU str: GTG

©IJRASET: All Rights are Reserved

codon: GTG str: GTG

codon: GTG str: CCC

codon: CCC str: TUC

codon: TUC str: CTC

codon: CTC str: GGC

codon: GGC str: GGC

codon: GGC str: GUG

codon: GUG str: CUG

codon: CUG str: UCT

codon: UCT str: UCC

codon: UCC str: G

READING FRAME 1 :

length:29

VSLARCPFGPGGVLSlen=87

str: uuC

codon: uuC str: Ctc

codon: Ctc str: tuu

codon: tuu str: CUU

codon: CUU str: TGC

codon: TGC str: UGG

codon: UGG str: CGC

codon: CGC str: GCT

codon: GCT str: UCT

codon: UCT str: CCG

codon: CCG str: TGT

©IJRASET: All Rights are Reserved

codon: TGT str: CCU

codon: CCU str: GCC

codon: GCC str: CCU

codon: CCU str: UCT

codon: UCT str: CCC

codon: CCC str: TGG

codon: TGG str: GUG

codon: GUG str: TGG

codon: TGG str: TGC

codon: TGC str: CCT

codon: CCT str: UCC

codon: UCC str: TCG

codon: TCG str: GCG

codon: GCG str: GCG

codon: GCG str: UGC

codon: UGC str: UGU

codon: UGU str: CTU

codon: CTU str: CCG

codon: CCG READING FRAME 2 :

length:29

FLWRPPAPPVSAACCPlen=86

str: uCC

codon: uCC str: tct

codon: tct str: uuC

codon: uuC str: UUT

©IJRASET: All Rights are Reserved

codon: UUT str: GCU

codon: GCU str: GGC

codon: GGC str: GCG

codon: GCG str: CTU

codon: CTU str: CTC

codon: CTC str: CGT

codon: CGT str: GTC

codon: GTC str: CUG

codon: CUG str: CCC

codon: CCC str: CUU

codon: CUU str: CTC

codon: CTC str: CCT

codon: CCT str: GGG

codon: GGG str: UGT

codon: UGT str: GGT

codon: GGT str: GCC

codon: GCC str: CTU

codon: CTU str: CCT

codon: CCT str: CGG

codon: CGG str: CGG

codon: CGG str: CGU

codon: CGU str: GCU

codon: GCU str: GUC

codon: GUC str: TUC

codon: TUC str: CG

©IJRASET: All Rights are Reserved

READING FRAME 3 :

length:28

SFAGALPLGARRRAVlen=88

str: GCC

codon: GCC str: UTC

codon: UTC str: UGU

codon: UGU str: CGU

codon: CGU str: GCG

codon: GCG str: GCG

codon: GCG str: GCT

codon: GCT str: CCU

codon: CCU str: TCC

codon: TCC str: CGT

codon: CGT str: GGT

codon: GGT str: GUG

codon: GUG str: GGT

codon: GGT str: CCC

codon: CCC str: TCU

codon: TCU str: UCC

codon: UCC str: CCG

codon: CCG str: UCC

codon: UCC str: TGT

codon: TGT str: GCC

codon: GCC str: TCU

codon: TCU str: TCG

©IJRASET: All Rights are Reserved

codon: TCG str: CGC

codon: CGC str: GGU

codon: GGU str: CGT

codon: CGT str: UUC

codon: UUC str: uut

codon: uut str: ctC

codon: ctC str: Cuu

codon: Cuu str: g

READING FRAME 4 :

length:29

ACRAAPVPSPSARGFLlen=87

str: CCU

codon: CCU str: TCU

codon: TCU str: GUC

codon: GUC str: GUG

codon: GUG str: CGG

codon: CGG str: CGG

codon: CGG str: CTC

codon: CUT str: CCC

codon: CCC str: GTG

codon: GTG str: GTG

codon: GTG str: UGG

codon: UGG str: GTC

codon: GTC str: CCT

codon: CCT str: CUU

©IJRASET: All Rights are Reserved

codon: CUU str: CCC

codon: CCC str: CGU

codon: CGU str: CCT

codon: CCT str: GTG

codon: GTG str: CCT

codon: CCT str: CUT

codon: CUT str: CGC

codon: CGC str: GCG

codon: GCG str: GUC

codon: GUC str: GTU

codon: GTU str: UCu

codon: UCu str: utc

codon: utc str: tCC

codon: tCC str: uug

codon: uug READING FRAME 5 :

length:29

PVVRRPWLPRRAVSLlen=86

str: CUT

codon: CUT str: CUG

codon: CUG str: UCG

codon: UCG str: UGC

codon: UGC str: GGC

codon: GGC str: GGC

codon: GGC str: TCC

codon: TCC str: UTC

©IJRASET: All Rights are Reserved

codon: UTC str: CCG

codon: CCG str: TGG

codon: TGG str: TGU

codon: TGU str: GGG

codon: GGG str: TCC

codon: TCC str: CTC

codon: CTC str: UUC

codon: UUC str: CCC

codon: CCC str: GUC

codon: GUC str: CTG

codon: CTG str: TGC

codon: TGC str: CTC

codon: CTC str: UTC

codon: UTC str: GCG

codon: GCG str: CGG

codon: CGG str: UCG

codon: UCG str: TUU

READING FRAME 6 :
length:28
LSCGGPGFPVARSLP

==========

****** WELCOME ****** TO ****** DNA FAM ****** WORLD ******

==========

!!*!*!* OPTIONS FOR UR DNA FAM FUNCTIONALITIES *!*!*!*!*

!!!

(1). Mutation Of DNA
(2). To Generate Random Sequence & Find Percentage Similarity
(3). To read DNA in Six Frame
(4). Exit

!!!

plz... enter ur choice ====> 4

==========

Sorry!!! plz.... Enter A Correct Choice

©IJRASET: All Rights are Reserved

********** THANK YOU FOR USING OUR SERVICE **********

==========

V. UTILITY AND FUTURE DEVELOPMENT

Empirical evidences of efficacy for calculating: (a) how mutation occurs randomly in DNA (a gene mutation), (b) demonstrating six possible open reading frames of DNA and (c) percentage identity between random DNA sequences, in the program DNA-FAM, would be helpful to identify areas for future research in providing a quick review on the DNA functionality for the benefit of scientific community and gene lovers.

DNA-FAM is the first approach as a free accessible academic real time programming software for DNA / gene codons worldwide. We are going to incorporate more scientific functionalities regarding DNA and gene functionalities as well as the complications arising due to DNA mutation. Periodically continuous updates shall be released to include other biological functionalities like, tRNA synthesis, the process of central dogma, to name a few. We plan to incorporate a provision to avail the required information using graphical user interface (GUI).

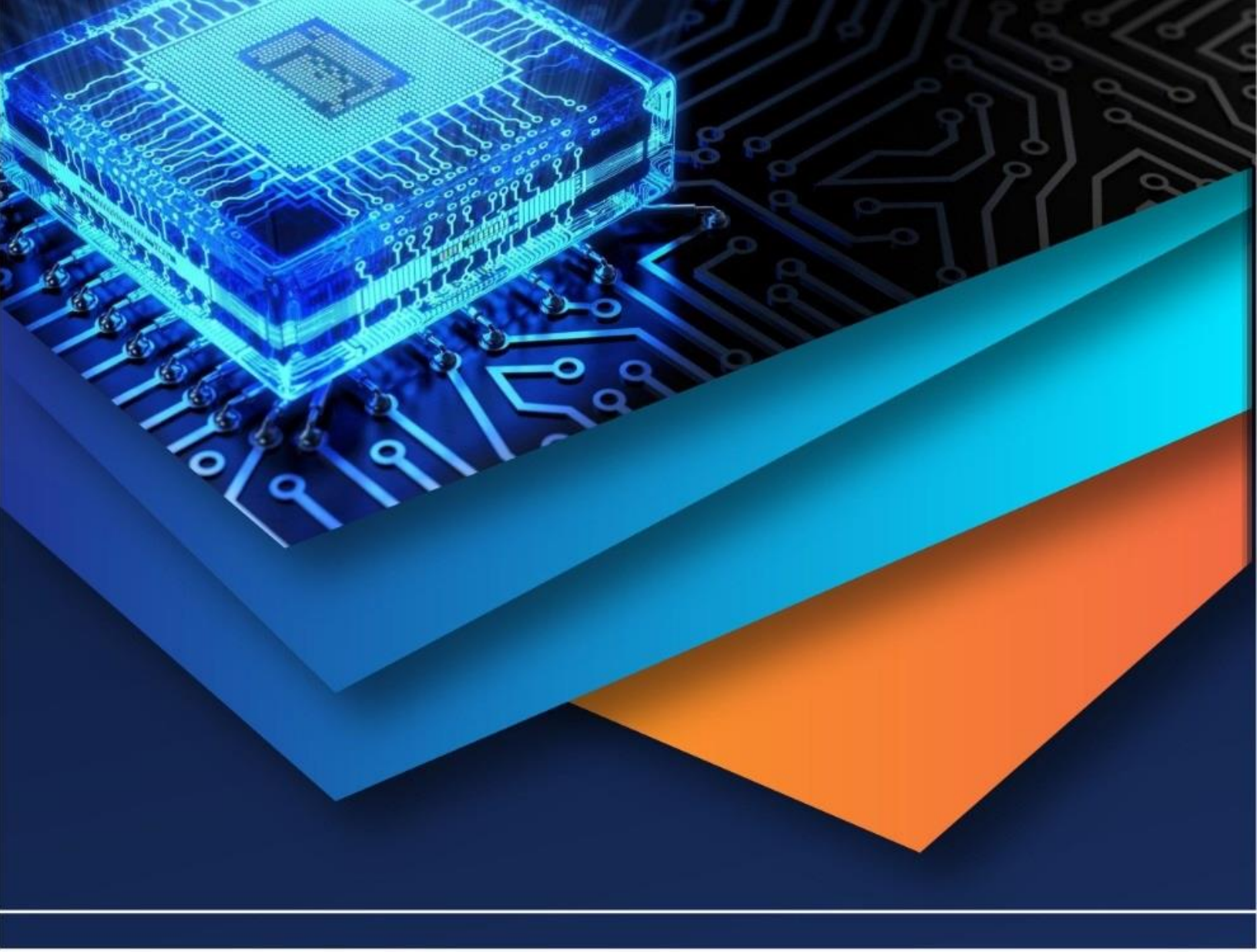
VI. ACKNOWLEDGMENT

We would like to acknowledge the scientific discussions that we had with Dr. Manjusha Nair from Amrita Vishwa Vidhyapeetham University, which helped in fine tuning of different programming modules.

REFERENCES

[1] R. Chandra, R. Rustgi, “Biodegradable Polymers,” Prog. Polym. Sci., vol. 23, pp. 1273-1335, Nov. 1998.
[2] J. D. Watson, F. H. Crick, “Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid,” Nature., vol. 171, pp. 737-738, Apr. 1953.
[3] E. D. Raczynska, M. Makowski, M. Hallmann, B. Kaminska, “Geometric and energetic consequences of prototrophy for adenine and its structural models – a review,” RSC Adv., vol. 5, pp. 36587-36604, Apr. 2015.
[4] S. Chatterjee, F. Wang, “How different is pyrimidine as a core component of DNA base from its diazine isomers: A DFT study?,” Int J Quantum Chem., vol. 116, pp. 1836-1845, Aug. 2016.
[5] H. C. Brown, E. A. Baude, F. C. Nachod, Determination of Organic Structures by Physical Methods, Academic Press, New York, 1955.
[6] T. L. Gilchrist, Heterocyclic Chemistry, New York: Longman, 1997.
[7] J. A. Joule, K. Mills, Heterocyclic Chemistry (5th ed.), Oxford: Wiley, 2010.
[8] L. Pauling, R. B. Corey, “A Proposed Structure for the Nucleic Acids”, Proc Natl Acad Sci U S A., vol. 39, pp. 84-97, Feb. 1953.
[9] M. F. Perutz, J. T. Randall, L. Thomson, M. H. Wilkins, J. D. Watson, “DNA helix”, Science., vol. 164, pp. 1537-1539, Jun. 1969.
[10] J. D. Watson, The Double Helix: A Personal Account of the Discovery of the Structure of DNA, Atheneum, 1980.
[11] M. H. Wilkins, A. R. Stokes, H. R. Wilson, “Molecular structure of deoxypentose nucleic acids”, Nature., vol. 171, pp. 738-740, Apr. 1953.
[12] W. Sanger, Principles of Nucleic Acid Structure, Springer-Verlag, 1984.
[13] K. Kruger, P. J. Grabowski, A. J. Zaug, J. Sands, D. E. Gottschling, T. R. Cech, “Self-splicing RNA: autoexcision and autocyclization of the ribosomal RNA intervening sequence of Tetrahymena”, Cell., vol. 31, pp. 147-157, Nov. 1982.
[14] J. S. Richardson, B. Schneider, L. W. Murray, G. J. Kapral, R. M. Immormino, J. J. Headd, D. C. Richardson, D. Ham, E. Hershkovits, L. D. Williams, K. S. Keating, A. M. Pyle, D. Micallef, J. Westbrook, H. M. Berman, “RNA Backbone: Consensus all-angle conformers and modular string nomenclature”, RNA., vol. 14, pp. 465-481, Mar. 2008.
[15] B. Alberts, A. Jhonson, J. Lewis, M. Raff, K. Roberts, P. Walter, Molecular Biology of the Cell; Chapter 5: DNA Replication, Repair, and Recombination, Garland Science, 2002.
[16] J. M. Berg, J. L. Tymoczko, L. Stryer, N. D. Clarke, Biochemistry; Chapter 27: DNA Replication, Recombination, and Repair, W. H. Freeman and Company, 2002.
[17] S. D. McCulloch, T. A. Kunkel, “The fidelity of DNA synthesis by eukaryotic replicative and translesion synthesis polymerases”, Cell Res., vol. 18, pp. 148-161, Jan. 2008.
[18] M. Kafri, E. Metzl-Raz, G. Jona, N. Barkai, “The cost of Protein Production”, Cell Rep., vol. 14, pp. 22-31, Jan. 2016.
[19] B. Alberts, Molecular biology of the cell, New York: Garland Science, 2002.
[20] B. Alberts, Molecular biology of the cell 5th Ed. New York: Garland Science, 2002.

©IJRASET: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)