DNA PATTERN DETECTION

FINAL REVIEW REPORT

Submitted by

Saravanan H	**18BCE2174**
Boyapati Sri Sai Ganesh	**19BCI0096**
Seru Santosh	**19BCI0133**
Prith Sharma	**19BCT0097**
Dinesh Kumar P	**19BDS0119**

Prepared For

Data Structures and Algorithms (CSE2003) – PROJECT COMPONENT

Submitted To

Dr. Dilip Kumar Choubey

Assistant Professor (Senior)

School of Computer Science and Engineering

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Table of Contents

1. Abstract

The project “DNA PATTEERN DETECTION” functions as follows: it inputs DNA sample's base data, compares them, and presents the percentage of similarity between them. It also can calculate the percentage of presence of particular DNA characteristics. This project carries a medical background usefulness and was primarily aimed to be launched in that field of use only.

2. Introduction:

2.1. Background:

Bioinformatics is the application of computer technology to the management and analysis of biological data. The result is that computers are being used to gather, store, analyze and merge biological data. The goal of bio-informatics is to uncover the wealth of biological information hidden in the mass of data and obtains a clearer insight into the fundamental biology of organisms. The most well known application of bioinformatics is sequence analysis. When we know a particular sequence is the cause for a characteristic, the trace of the sequence in the DNA and the number of occurrences of the sequence defines the intensity of the characteristic.

2.2. Motivation

Every human has genes, these genes are made up of DNA(Deoxyribonucleic Acid), and the DNA sequence of each human is unique. Yet the DNA sequences of all individuals are strikingly 99.9% similar, indicating that the difference is just 0.1%. DNA is found in every living cell in a body and is the carrier of the genetic code of the organism. The genetic code is information, which will be needed for biological growth and reproductive inheritance. Biologists can use comparisons of DNA to discover evolutionary difference, the causes of disease and how genetic codes are applied between different organisms. Usually a particular disease has a particular DNA pattern and if we identify the pattern, we will be able to give a proper diagnosis to the patient. Also with the help of DNA comparisons, we can compare the DNA of the parents and the children and if the parent DNA has some genetic defects, there is at least a 50% chance of a child getting that disease and we can treat the child earlier.

2.3. Objective

KMP algorithm is proven the most efficient method in pattern matching with the lowest time complexity and with an acceptable space complexity from comparisons made from various researches. The objective of the project is to construct a KMP algorithm, which could input the DNAs, compare the DNAs and detect the presence of some characteristics.

2.4. Contributions of the project:

This project carries a medical background usefulness and was primarily aimed to be used in that field of use. This project will hold paramount significance for the genetics department. It can be helpful especially to find what kind of characteristics the DNA might carry and help the patients get the proper diagnosis based on this detection.

3. Project Resource Requirements:

3.1. Software Requirements:

- C/CPP IDE
- GNU GCC compiler
- Operating system – Windows 10/8 /7/XP

3.2. Hardware Requirements:

- Intel Pentium IV or higher
- 2 GB RAM or higher
- 10 MB of free space

4. Literature Survey

4.1. Background:

Exact sequence matching is a vital component of many problems, including text editing, information retrieval, signal processing and recently in bioinformatics applications. Various sequence matching or string matching algorithms are found in literature. The reliability of these algorithms constantly depends on the ability to detect the presence of match characters and the ability to discard any mismatch characters. These algorithms are widely used for searching of an unusual sequence in a given DNA sequence. The objective of the literature survey is to find the most efficient method for DNA pattern detection.

4.2. Literature Review

Authors	Method	Purpose	Advantages	Disadvantages
Mohammad Riyaz Hossen,Md. Shafiul Azam,Humayan Kabir Rana[1]	Comparing the Naive exact matching algorithm, Boyer-Moore Algorithm and Knuth-Morris-Pratt Algorithm (KMP) using Human hair, Leukemia virus and HIV virus datasets	To compare the performance of Naïve exact matching, Boyer-Moore (BM) and Knuth-Morris-Pratt (KMP) algorithms for DNA pattern Matching	1.The literature was able to find the most efficient algorithm. 2.KMP algorithm was found to be the most simple algorithm and the Boyer-Moore algorithm was the most efficient.	Boyer-Moore Algorithm was better than KMP by comparing their best cases. The worst case of the Boyer-Moore algorithm was worse than the KMP algorithm.

S. Rajesh,S. Prathima,Dr.L.S.S. Reddy[2]	The number of occurrences a particular pattern in DNA signifies the intensity of a particular disease like cancer. The literature uses Brute force algorithm, Boyer-Moore Algorithm and KMP algorithm to detect unusual patterns using gene database	To propose a String and Pattern matching algorithms to find out a particular sequence in the given DNA.	1. The brute force algorithm obviously was not able to perform efficiently, it took O(nm) time. 2. The Boyer-Moore algorithm worked the fastest for moderately sized alphabets but failed to keep up with the same efficient for larger sized alphabets 3. KMP algorithm was found to be the better of all three with O(n+m) time complexity	-

Nitashi Kalita, Chitra, Radhika Sharma and Samarjeet Borah[3]	The traditional KMP algorithm searches for the exact sequence (short sequence) that can be present in the longer sequence and overlooks the pattern that can have few mismatches and this cannot be overlooked.The authors of this literature proposed a new method for enhancing the KMP algorithm using a window and a percentage of mismatch.	To propose a modified KMP algorithm for searching for an unusual sequence in a given DNA sequence.	The enhanced KMP algorithm was more complete than the traditional algorithm. It's more appropriate for DNA.	1.The traditional KMP algorithm had a better time complexity than the enhanced KMP algorithm. 2. Scalability issues were not considered

NYO ME TUN AND THIN MYA MYA SWE[4]	The main process of this system is to find the matched DNA sequence in the DNA database that finds the matched DNA sequence in the DNA database using these pattern matching algorithms. This system compares similarity values with threshold values and stores particular results which are diseased or not.	To find the matched DNA sequence in the DNA database that finds the matched DNA sequence in the DNA database using these pattern matching algorithms.	Comparing three pattern matching algorithms,this system can identify optimal results according to each result using Java programming language and comparison results are demonstrated with a bar graph and processing time of each algorithm.	Only one disadvantage On this process that is it will take more time as Comparing of three pattern matching algorithm, Boyer-Moore Algorithm,Brute-Force Algorithm and Knuth-Morris-Pratt Algorithm (KMP)

Izzat Alsmadi and Maryam Nuser[5]	This paper evaluates two algorithms used for DNA comparison. Those are: Longest Common Substring and Subsequence (LCS, LCSS). Evaluation is performed based on the different code implementations for those two algorithms. Accuracy and performance are the two major criteria that are used for the evaluation of algorithms' implementation.	To compare two algorithms used for DNA comparison. Those are: Longest Common Substring and Subsequence (LCS, LCSS).	1.)Using this we know which algorithm is better. 2.)LCSS calculation takes longer time or exhausts system memory and causes a crash and hence their values were excluded. in comparison to LCS 3)Another finding is that time is not perfectly increasing with the increase in the size of the sequence.	In the same DNA sequences on different tools may show different results. While some of the differences are shown to be expected and are part of the different default considerations or interpretations of those algorithms, other results showed that implementations for the same algorithm are somewhat different and inconsistent.

T. M. Inbamalar and R. Sivakumar[6]	.In this paper, we discuss methods based on wavelet transformation of DNA sequences to identify protein coding regions In DNA sequences.The main aim of this paper is to increase the accuracy and also to reduce the noise as much as possible.	To identify protein coding regions In DNA sequences.	The results showed that our method has ability to detect even the short coding region and it outperforms existing methods.	It is difficult to detect the first coding region along the positions 929–1039 properly in the DFT spectral content method and antinotch filter.
M Yazid,M Nordin,R Rahaman,A Azad[7]	The DNA alphabet consists of the four nucleotides a, c, g and t (standing for adenine, cytosine, guanine, and thymine, respectively) used to encode DNA, and could be signed as Σ = {a, c, g, t}. Normally, when a new DNA or protein sequence is determined, it would be compared to all known sequences in the annotated	DNA sequence alignment for similarity search is a vital topic in bioinformatics algorithm development. Computational searching for a set of DNA sequences, S, that similar to a query sequence, q, in a large scale of DNA databases is very complicated and requires high processors performance as well as large memory spaces.	1)The use of parallel computers to speed the processing of those phases. 2)To increase the sensitivity of filtering mechanism, we are concern to focus on the problem of pattern matching that allows errors (approximate string matching).	1)Trying to use another approach in filtering processing such as Aho-Corasick automaton machine and its variants.

	databases such as GenBank, SwissProt and EMBL. The paper demonstrates the employ of exact string matching KMP algorithm which has time complexity O(m+n) as a filtration mechanism before an optimal alignment (Smith-Waterman algorithm) between sequences is implemented. Based on the propose framework, a tool has been developed using object oriented programming language Java			
Robert M Zink, George F Barrowclough **[8]**	Mitochondrial DNA is present in most cells in high copy number and is relatively easy, rapid, and inexpensive to sequence. For example, if well-preserved or fresh tissue samples are	Mitochondrial DNA (mtDNA) has been the workhorse of research in phylogeography for almost two decades. However, concerns with basing evolutionary interpretations	1)the former ought to be taken as indicative of lineage divergence. 2)modern *evolutionary genetics* studies that wish to describe both phylogeographic patterns and	1)We strongly urge that nuclear sequences, and not frequency-based approaches such as microsatellites, complement mtDNA data for those cases.

	available, then sequences of a kilobase or more can be produced for 100 or more individuals, perhaps distributed over 10 or more populations (limitations stem mainly from costs and time to gather samples).	on mtDNA results alone have been voiced since the inception of such studies. Recently, some authors have suggested that the potential problems with mtDNA are so great that inferences about population structure and species limits are unwarranted unless corroborated by other evidence, usually in the form of nuclear gene data	evolutionary processes will require multiple loci	
Sajid A Marhon, Stefan C Kremer[9]	These DSP techniques rely on the phenomenon that protein-coding regions have a prominent power spectrum peak at frequency $f = \frac{1}{3}$ arising from the length of codons (three nucleic acids).	The identification of regions of DNA sequences that code for proteins is one of the most fundamental applications in bioinformatics. These protein-coding regions are in contrast to other DNA regions that encode functional RNA molecules, provide structural	1)Based on this partitioning, DSP techniques can be easily described and compared based on their unique implementations of the processing steps. 2) We compare the approaches, and discuss strengths and weaknesses of each in the context of different applications.	1) It an accessible introduction and comparative review of DSP methods for the identification of protein-coding regions.

		stability of chromosomes, serve as genetic raw materials, represent molecular fossils, or have no known purpose (sometimes called “junk DNA”)		
Pandiselvam. P, Marimuthu. T, Lawrance. R **[10]**	In this paper, different kinds of string matching algorithms were studied and their time and space complexities were observed. For this study, we have assessed the performance of algorithms tested with biological sequences.	To study the various aspects of different algorithms such as KMP, Hamming, Brute Force, Rabin Karp, etc.for proposing a speedy and efficient string matching algorithm for biological sequences.	1. The literature was able to find a couple of algorithms which would be a good choice for working on biological specimens. 2. KMP algorithm relatively easier to implement because never needs to move backwards in the input sequence.	A

Samarjeet Borah, Debashree Bhattacharjee, Krishna Vijay	This project implements KMP algorithm by using the	To draw a clear comparison between our KMP algorithm	1. The searching speed of the	1. The Multithreaded KMP requires high

Kr. Singh, Bikash Rai **[11]**	concepts of multithreading and divide and conquer. In this research work a study on the KMP algorithm is done which is a fast string matching algorithm that is primarily used for matching huge disease DNA sequence or any huge pattern of DNA sequence with a given source DNA sequence	compared to the multithreaded KMP algorithm.	multithreaded algorithm is much faster than the normal KMP algorithm. 2.This application can be used to match patterns of various strings of any huge length to find their relative percentages and positions.	speed processors to produce the efficient results and so the implementation was carried out at HP-Z400 Workstation. 2.It can work accurately only on large text strings and pattern strings. 3.The application can be used only in operating systems that support thread applications. 4.In the experiments carried out it is found that space complexity of the application is really high so future improvements can be made on this area to reduce this requirement.

Prof. I.V.Srinivas,	This studies various algorithms	To study the different string	1.KMP algorithm can	1. This suggests that Boyer-

Moez Samnani, Mohammed Shafaat Shaikh **[12]**	for the String matching issue. For the string matching problem, we assume that the text is in array T [1...n] and n is length of text array and pattern is in array [1...m]and m is length of pattern (m<=n).	matching algorithms.In this paper, various types of string matching algorithms were examined with organic arrangements for example, DNA and Proteins.	work quite well, if our alphabet is small (e.g. DNA bases), as we get a higher chance that our search patterns contain reusable sub-patterns.	Moore is the best algorithm to tackle string matching in regular cases. But, in reality, there is no definite answer to the overall best. It is rather a matter of choosing the right tool for the pattern at hand.
Juan V Lorenzo-Ginori, Aníbal Rodríguez-Fuentes, Ricardo G Abalo, Robersy Sanchez Rodriguez[13]	The use of DSP principles to analyze genomic sequences requires defining an adequate representation of the nucleotide bases by numerical values, converting the nucleotide sequences into time series. Once this has been done, all the mathematical tools usually employed in DSP are used in solving tasks such as identification of protein coding DNA regions.	The detection of tandem repeats is important in biology and medicine as it can be used for phylogenetic studies and disease diagnosis.	Significant peaks in the spectrogram are selected, and the corresponding regions in the DNA sequence are analyzed to search for tandem repeats. Experiment results show that the method has a superior performance in comparison with other algorithms.	-
Haruo Ohmori, Jun-ichi Tomizawa,	Col E1 DNA has methylated cytosine in the sequence 5′-CC*(A/T)GG-3′	The existence of the methylated cytosine can be confirmed by analyzing the	It reduces the effect of background noise in the period-3	The proposed algorithm leads to decrease in the computational

Allan M Maxam[14]	and methylated adenine in the sequence 5′-GA*TC-3′ at the positions indicated by asterisks(*). When the Maxam-Gilbert DNA sequencing method is applied to this DNA, the methylated cytosine (5-methylcytosine) is found to be less reactive to hydrazine than are cytosine and thymine, so that a band corresponding to that base does not appear in the pyrimidine	complementary strand or unmethylated DNA. In contrast, the methylated adenine	spectrum, we used the discrete wavelet transform at three levels and applied it on the input digital signal. Finally, the Goertzel algorithm was used to extract period-3 components in the filtered DNA sequence	complexity and hence, increases the speed of the process. Detection of small size exons in DNA sequences, exactly, is another advantage of the algorithm.
David Posada, Keith A Crandall[15]	Recombination is a key evolutionary process that shapes the architecture of genomes and the genetic structure of populations. Although many statistical methods are available for the detection of recombination from DNA sequences, their absolute and relative	In this paper, an efficient algorithm for tandem repeat detection is proposed. In our method, the spectrogram of a DNA sequence is analyzed based on the autoregressive model.	As they develop, these analyses will no doubt provide significant advances in the field of restoration ecology and the identification of appropriate locations for species reintroduction, as well as highlighting	The approach is based on the mapping of DNA symbols to pure quaternions. The resulting quaternionic periodicity transform does not outperform the previously proposed complex periodicity transform due to enhanced,

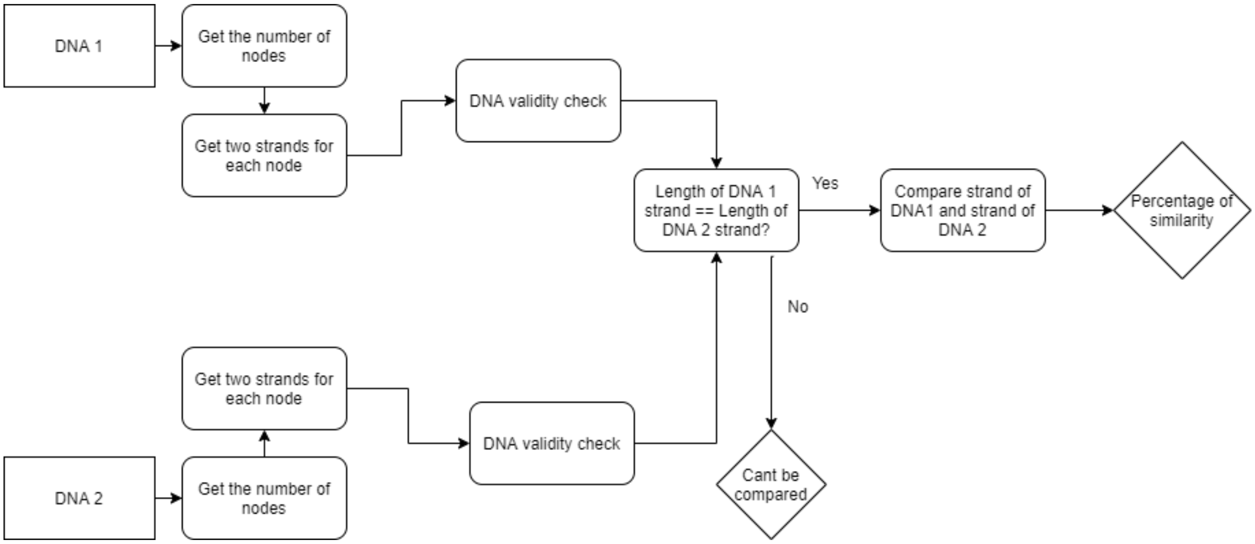
	performance is still unknown.		species at ecological risk	symbol-balanced sensitivity to DNA patterns.

4.3. Summary:

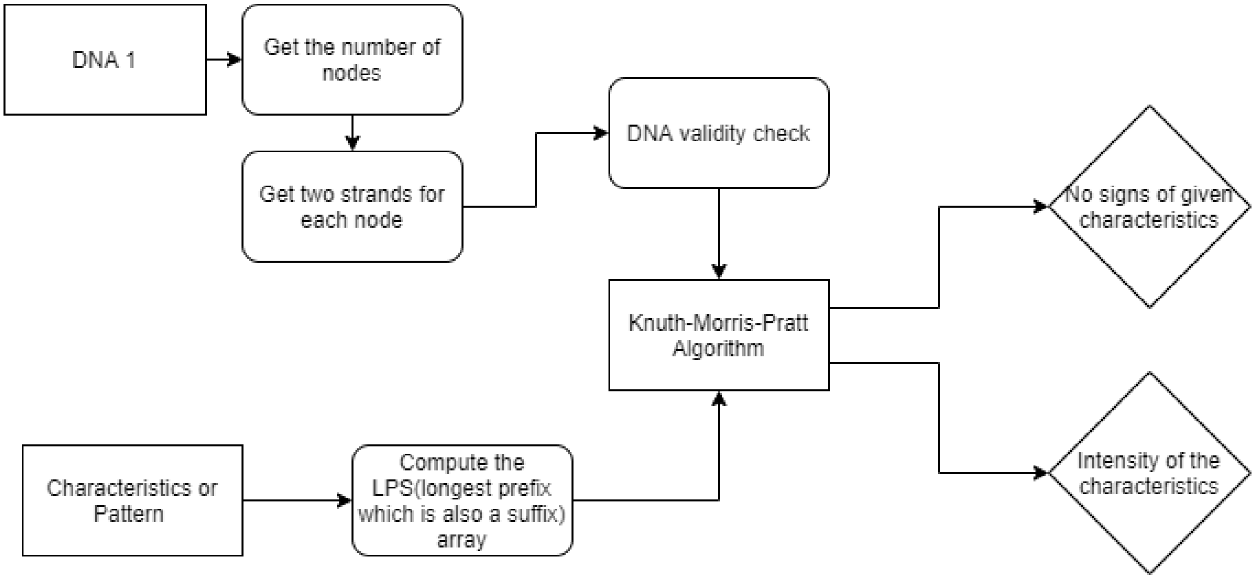
From the literature review of 4 papers, Knuth-Morris-Pratt algorithm was found to be the best algorithm for DNA pattern detection especially because due to the repetitive characters of the DNA strands. The KMP has large advantages that it is guaranteed worst case efficient and it is only required to parse the entire string once. The preprocessing time is always O(n) and the searching time is always O(m) where n is the length of the pattern and m is the length of the DNA strand. But it is important to note that KMP algorithm wouldn’t not be the most efficient algorithm if there are not many overlapping part of the string.

5. Proposed Architecture:

DNA comparison:


DNA 1
Get the number of nodes
Get two strands for each node
DNA validity check
Length of DNA 1 strand == Length of DNA 2 strand?
Yes
Compare strand of DNA1 and strand of DNA 2
Percentage of similarity
No
Cant be compared
Get two strands for each node
DNA validity check
DNA 2
Get the number of nodes

DNA Characteristics detection:


DNA 1
Get the number of nodes
Get two strands for each node
DNA validity check
Knuth-Morris-Pratt Algorithm
No signs of given characteristics
Intensity of the characteristics
Characteristics or Pattern
Compute the LPS(longest prefix which is also a suffix) array

6. Implementation:

6.1 Introduction:

This project has implemented the DNA pattern matching using KMP algorithm due to its efficiency. It has the worst time complexity O(n). The basic idea behind KMP's algorithm is: whenever we detect a mismatch (after some matches), we already know some of the characters in the text of the next window. We take advantage of this information to avoid matching the characters that we know will anyway match.

Deoxyribonucleic acid is a thread-like chain of nucleotides carrying the genetic instructions used in the growth, development, functioning and reproduction of all known living organisms and many viruses. Bases: The rules of base pairing (or nucleotide pairing) are:

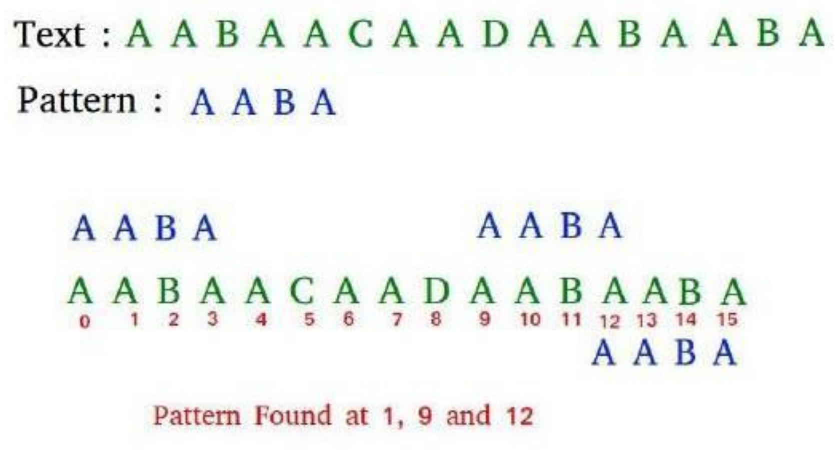
- A with T: the purine adenine (A) always pairs with the pyrimidine thymine (T)
- C with G: the pyrimidine cytosine (C) always pairs with the purine guanine (G)

DNA Comparison: The project inputs DNA sample's base data and compares them and presents the percentage of similarity between them.
DNA Analysis: The project can calculate the percentage of presence of a particular DNA characteristics. The KMP algorithm is used to match the characteristics with the DNA sample and then the percentage is presented.

KMP ALGORITHM INTRODUCTION:

KMP(Knuth Morris Pratt)PatternSearching
The Naive pattern searching algorithm doesn't work well in cases where we see many matching characters followed by a mismatching character. Following are some examples.
The KMP matching algorithm uses degenerating property (pattern having same sub-patterns appearing more than once in the pattern) of the pattern and improves the worst case complexity to O(n). The basic idea behind KMP's algorithm is: whenever we detect a mismatch (after some matches), we already know some of the characters in the text of the next window. We take advantage of this information to avoid matching the characters that we know will anyway match.

OVER VIEW:
Input:
txt[] = "THIS IS A TEST TEXT"
pat[] = "TEST"
Output: Pattern found at index 10
txt[] = "AABAACAADAABAABA"
pat[] = "AABA" Output: Pattern found at index 0
Pattern found at index 9 Pattern found at index 12


Text : A A B A A C A A D A A B A A B A
Pattern : A A B A
A A B A
A A B A
A A B A A C A A D A A B A A B A
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
A A B A
Pattern Found at 1, 9 and 12

6.2. Pseudocode for the KMP algorithm:

computeLSParray:

```
Begin
  length := 0
  prefArray[0] := 0

  for all character index ‘i’ of pattern, do
    if pattern[i] = pattern[length], then
      increase length by 1
      prefArray[i] := length
    else
      if length ≠ 0 then
        length := prefArray[length - 1]
        decrease i by 1
      else
        prefArray[i] := 0
  done
End
```

KMPAlgorithm:

```
Begin
   n := size of text
   m := size of pattern
   call findPrefix(pattern, m, prefArray)

   while i < n, do
      if text[i] = pattern[j], then
         increase i and j by 1
      if j = m, then
         print the location (i-j) as there is the pattern
         j := prefArray[j-1]
      else if i < n AND pattern[j] ≠ text[i] then
         if j ≠ 0 then
            j := prefArray[j - 1]
         else
            increase i by 1
   done
End
```

6.3. Code:

Code:

```
#include<stdio.h>
#include<iostream>
#include<stdlib.h>
#include<conio.h>
#include<bits/stdc++.h>
using namespace std;
int main_page();
int abt_page();
int welcomepg();
int instruct();
int FAQ();

void computeLPSArray(char pat[], int M, int lps[]);
void KMPSearch(char* pat, char* txt, int M, int N);
void rundpg();
void input();
void input1();
void comparing(char D1S1[], int count1, char D2S1[], int count2);

int inputcheck(char D1S1[], char D1S2[], int count);
int welcomepg(){
    system("cls");
    cout<<" \n \n \n \n"<<"\t\t\t\t\t\t "<<" WELCOME \n"<<"\t\t\t\t\t\t"<<" TO \n"<<"\t\t\t\t\t"<<"** DNA PATTERN DETECTION **";
    getch();
    main_page();
}

int abt_page(){
    system("cls");
    cout<<" \n "<<"\t\t\t\t\t\t\t\t\t\t"<<"DNA PATTERN DETECTION";
    cout<<"\nABOUT PAGE";
    cout<<"\nTwo tasks can be performed:"<<endl;
```

```
int main_page(){
    system("cls");
    int n;
    cout<<"\n"<<"\t\t\t\t\t\t\t\t\t\t"<<"DNA PATTERN DETECTION";
    cout<<"\n1.About Us"<<endl<<"2.Instructions"<<endl<<"3.FAQ"<<endl<<"4.Run"<<endl<<"5.Exit"<<endl;
    cout<<"Choose an option to proceed:";
    cin>>n;
    switch(n){
        case 1:
            int a1;
            a1=abt_page();
            if(a1==1){
                main_page();
            }
            else if(a1==2){
                return 0;
            }
            else{
                cout<<"invalid input";

            }
            break;
        case 2:
            int a2;
            a2=instruct();
            if(a2==1){
                main_page();
            }
            else if(a2==2){
                return 0;
            }
            else{
                cout<<"invalid input";
            }
            break;
        case 3:
            int a3;
            a3=FAQ();
            if(a3==1){
                main_page();
    cout<<"\n\t1.To check the percent
matching of 2 DNA of same or different
species"<<endl;
    cout<<"\t2.To check the percentage
of characteristics present in the
particular DNA\n\n";
    cout<<"Go to:";
    cout<<endl<<"\t1.Main
page"<<endl<<"\t2.Exit";
    cout<<"\nChoose an option to
proceed:";
    int n;
    cin>>n;
    return n;
}
int instruct(){
    system("cls");
    cout<<" \n
"<<"\t\t\t\t\t\t\t\t\t"<<"DNA
PATTERN DETECTION";
    cout<<"\n INSTRUCTIONS
PAGE\n";
    cout<<"\n INPUT:\n"<<"\n\tTHE
DNA MUST BE INPUTTED BASE BY
BASE, EACH IN CAPITALS. THEY
ARE STORED AS CHAHRACTER
ARRAY, NOT AS STRING SO USER
MUST ENTER ONE BASE AT A
TIME.\n";
    cout<<"\n OUTPUT: \n"<<"\n\tIN
THE COMPARING FUNCTION, WE
COMPARE THE TWO INPUTTED
DNAS AND OUTPUT THE
PERCENTAGE OF DNA
MATCHED."<<"\n\tFOR EXAMPLE
IF THE OUTPUT SAYS 25% MATCH,
IT MEANS THAT 25% OF DNA 1 IS
PRESENT IN DNA 2. HENCE THE
SPECIES 2 WILL HAVE 25%
CHARACTERISTICS OF SPECIES
1.";
    cout<<"\n\tIN THE MATCHING
CHARACTERISTICS FUNCTION,
WE FIND THE % OF THE INPUT
CHARACTERISTIC IN THE INPUT
DNA."<<"\n\tFOR EXAMPLE IF WE
```

```
            }
            else if(a3==2){
                return 0;
            }
            else{
                cout<<"invalid input";

            }
            break;
        case 4:
            rundpg();
            break;
        case 5:
            return 0;
        default:
            cout<<"Invalid input\n" ;
    }
    return 0;
}

void input(){
    system("cls");
    int i, m, count1=0, ch, len;
    char S1[80], S2[80], txt[35];
    cout<<" \n "<<"\t\t\t\t\t\t\t\t\t"<<"DNA
PATTERN DETECTION";
    cout<<"\n INPUT PAGE";
    cout<<"\n Enter the number of nodes in the
DNA: ";
    cin>>m;
    count1=m*10;
    cout<<" Enter the first strand of the DNA:";
    for(i=0;i<count1;i++){
        cin>>S1[i];
    }
    cout<<" Enter the second strand of the DNA:";
    for(i=0;i<count1;i++){
        cin>>S2[i];
    }
    ch=inputcheck(S1,S2, count1);
    if(ch==1){
        cout<<"\n The first strand of the DNA : \n ";
        for(i=0;i<count1;i++){
            cout<<S1[i]<<" ";
        }
        cout<<"\n The second strand of the DNA :\n ";
```

```
SAY THAT THERE IS 20%
MATCHING OF THE
CHARACTERISTICS IN THE DNA,
THEN WE CAN SAY THAT THE
SPECIES HAS 20% OF THAT
CHARACTERISTIC.";
    cout<<"\n\nGo to:";
    cout<<endl<<"\t1.Main
page"<<endl<<"\t2.Exit";
    cout<<"\nChoose an option to
proceed:";
    int n;
    cin>>n;
    return n;
}

int FAQ(){
    system("cls");
    cout<<" \n
"<<"\t\t\t\t\t\t\t\t\t\t"<<"DNA
PATTERN DETECTION";
    cout<<"\n FAQ PAGE\n \n";
    cout<<"Here are some general
questions which arise about
DNA\n\n"<<endl;
    cout<<"1. What is a strand of
DNA?\n"<<endl;
    cout<<"A strand of DNA is simply a
collection of deoxyribonucleotides(the
monomers of a DNA molecule),
sometimes called a fragment of
DNA\n\n"<<endl;
    cout<<"2.How many strands make up
a DNA double helix?\n"<<endl;
    cout<<"A double helix of DNA is
formed when two DNA strands each
having a 5&#39; and 3&#39; end wind
around each other. Each of the DNA
strands have nucleotides present.The
two strands are held by hydrogen
bonds.\n\n"<<endl;
    cout<<"3. What is the double
helix?\n"<<endl;
    cout<<"It's the shape of the DNA
molecule, a pair of parallel helices
```

```
        for(i=0;i<count1;i++){
            cout<<S2[i]<<" ";
        }
    }
    if(ch!=1){
        cout<<"\n The inputed DNA were incorrect.
Kindly re-input.";
        input();
    }
    else{
        cout<<"\n Enter the length of the
characteristics to the compared with the DNA
sample : ";
        cin>>len;
        cout<<"\n Enter the characteristics to the
compared with the DNA sample\n";
        for(i=0;i<len;i++){
            cin>>txt[i];
        }
        getch();
        KMPSearch(txt,S1, len, count1);
    }
}

void input1(){
    system("cls");
    int i, m1, m2, count1=0, count2=0, ch1, ch2;
    char D1S1[80], D1S2[80], D2S1[80], D2S2[80];
    cout<<" \n "<<"\t\t\t\t\t\t\t\t\t\t"<<"DNA
PATTERN DETECTION";
    cout<<"\n INPUT PAGE";
    cout<<"\n Enter the number of nodes in the DNA
1: ";
    cin>>m1;
    count1=m1*10;
    cout<<" Enter the first strand of the DNA 1:";
    for(i=0;i<count1;i++){
        cin>>D1S1[i];
    }
    cout<<" Enter the second strand of the DNA 1:";
    for(i=0;i<count1;i++){
        cin>>D1S2[i];
    }
    ch1=inputcheck(D1S1,D1S2, count1);
    cout<<"\n Enter the number of nodes in the DNA
2: ";
```

```
intertwind about a common
axis.\n\n"<<endl;
  cout<<"4. Why are the two strands of
the double helix described as
complementary?\n"<<endl;
  cout<<"Well, because the sequence
of bases on one strand determines the
sequence of bases on the other
strand.\n\n"<<endl;
  cout<<"5.Why are DNA strands
within the double helix said to be
complimentary?\n"<<endl;
  cout<<"DNA strands are said to be
complementary because they both match
up with each other; A with T and C with
G. So if you have the strand ATGGCTA
the complementary strand (the other half
of the double helix) would read
TACCGAT. So if you know one side of
the strand then you can describe the
whole.\n\n"<<endl;
  cout<<"6.One reason DNA chains
twist into a double helix is for the
purpose of what?\n"<<endl;
  cout<<"DNA chain twists so that the
bases are closer together in the double
helix. The DNA chain also takes up less
space this way.\n\n"<<endl;
  cout<<"7.Each half of the DNA
molecule is a DNA strand why is DNA
also called a double helix?\n"<<endl;
  cout<<"Ok each half of a DNA
molecule is not DNA it is RNA which is
a single strand of DNA!\n\n"<<endl;
  cout<<"8.Why is RNA only
synthesized from one strand of a
doublestranded DNA helix?\n"<<endl;
  cout<<"Not sure but I'm going to
make an educated guess, your DNA
doesn't want to leave the nucleus so it
stays protected, your RNA is a translator
and messenger it already has the 1
strand that can cooperate with the DNA
so it can copy it exactly and then
translate it.\n\n"<<endl;
  cout<<"Go to:";
```

```
  cin>>m2;
  cout<<" Enter the first strand of the DNA 2:";
  count2=m2*10;
  for(i=0;i<count2;i++){
    cin>>D2S1[i];
  }
  cout<<" Enter the second strand of the DNA 2:";
  for(i=0;i<count2;i++){
    cin>>D2S2[i];
  }
  ch2=inputcheck(D2S1,D2S2, count2);
  if(ch1==1){
    cout<<"The first strand of the DNA 1 :";
    for(i=0;i<count1;i++){
      cout<<D1S1[i]<<" ";
    }
    cout<<"The second strand of the DNA 1:";
    for(i=0;i<count1;i++){
      cout<<D1S2[i]<<" ";
    }
  }
  if(ch2==1){
    cout<<"\n The first strand of the DNA 2 : \n ";
    for(i=0;i<count2;i++){
      cout<<D2S1[i]<<" ";
    }
    cout<<"\n The second strand of the DNA 2:\n
";
    for(i=0;i<count2;i++){
      cout<<D2S2[i]<<" ";
    }
  }
  if(ch1!=1 || ch2!=1){
    cout<<"\n The inputed DNA were incorrect.
Kindly re-input.";
    input1();
  }
  getch();
  comparing(D1S1, count1, D2S1, count2);
}

void comparing(char D1S1[], int count1, char
D2S1[], int count2){
  system("cls");
  int i, a=0,b=0;
  double n,m;
```

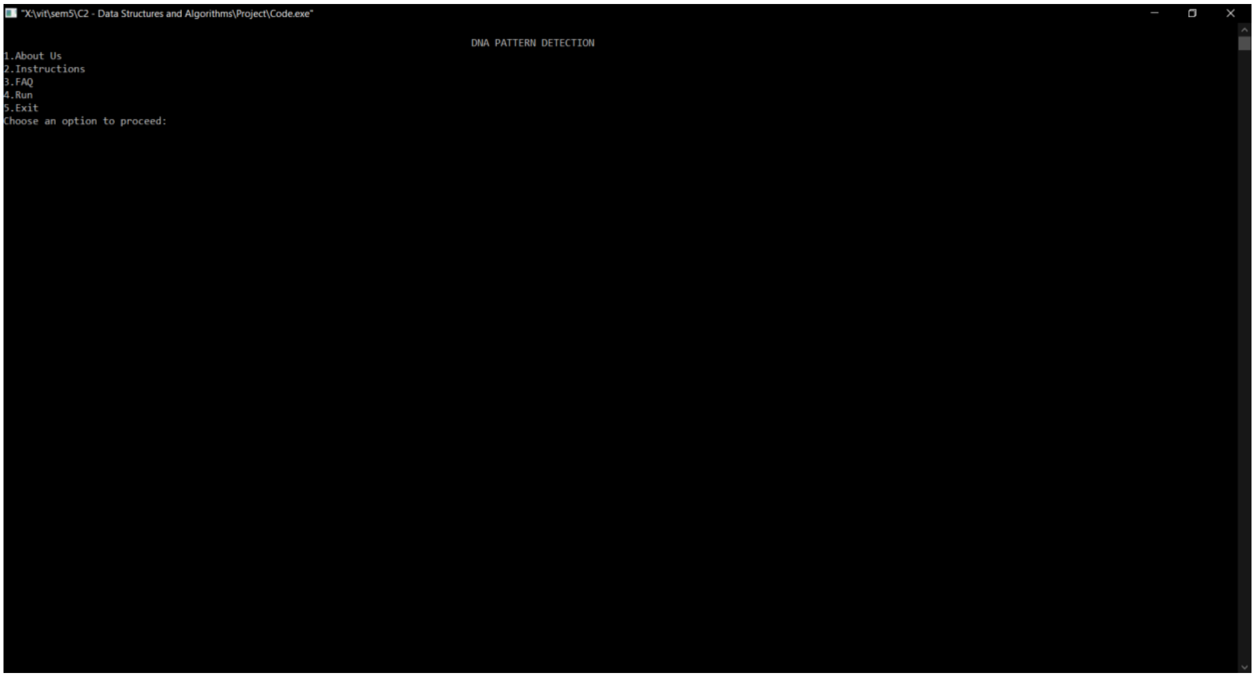
```
    cout<<endl<<"\t1.Main
page"<<endl<<"\t2.Exit";
    cout<<"\nChoose an option to
proceed:";
    int n;
    cin>>n;
    return n;
}

void KMPSearch(char* pat, char* txt,
int M, int N){
    system("cls");
    int counter=0;
    int lps[M];
    cout<<" \n
"<<"\t\t\t\t\t\t\t\t\t"<<"DNA
PATTERN DETECTION\n\n";
    cout<<"\n\n RESULTS OF THE
CHARACTERISTICS COMPARISON
OF THE DNA SAMPLE";
    computeLPSArray(pat, M, lps);
    int i = 0;
    int j = 0;
    while (i < N){
        if (pat[j] == txt[i]){
            j++;
            i++;
        }
        if (j == M){
            counter++;
            j = lps[j-1];
        }
        else if (i < N && pat[j] != txt[i]){
            if (j != 0){
                j = lps[j-1];
            }
            else{
                i = i+1;
            }
        }
    }
    if(counter==0){
        cout<<"\n\n THE GIVEN
CHARACTERISTIC COULD NOT BE
FOUND"<<endl;
    }
```

```
    cout<<" \n "<<"\t\t\t\t\t\t\t\t\t"<<"DNA
PATTERN DETECTION";
    cout<<"\n RESULTS OF DNA SAMPLE
COMPARISON";
    if(count1!=count2){
        cout<<"\n The two DNA's cannot be
compared as the strand lengths vary.";
    }
    else{
        for(i=0; i<count1;i++){
            if(D1S1[i]==D2S1[i]){
                a++;
            }
            else{
                b++;
            }
        }
        n=(float)a/(float)(a+b);
        m=n*100;
        cout<<"\n\n\nFrom the comparison of the two
DNA samples, it is identified that the percentage of
similarity between the two DNA's
are"<<endl<<m<<"%";
    }
}
int inputcheck(char D1S1[], char D1S2[], int
count){
    int i, a=0, b=0, m=0;
    for(i=0;i<count;i++){
        if(D1S1[i]=='A'){
            if(D1S2[i]=='T'){
                a++;
            }
            else{
                b++;
                break;
            }
        }
        else if(D1S1[i]=='T'){
            if(D1S2[i]=='A'){
                a++;
            }
            else{
                b++;
                break;
            }
    else{
        double per=(counter*M*100)/N;
        cout<<"\n THE DNA HAD
"<<per<<"%
CHARACTERISTICS"<<endl;
    }
}

void computeLPSArray(char pat[], int
M, int lps[]){
    int len = 0;
    lps[0] = 0;
    int i = 1;
    while (i < M){
        if (pat[i] == pat[len]){
            len++;
            lps[i] = len;
            i++;
        }
        else{
            if (len != 0){
                len = lps[len-1];
            }
            else{
                lps[i] = 0;
                i++;
            }
        }
    }
}

void rundpg(){
    system("cls");
    cout<<" \n
"<<"\t\t\t\t\t\t\t\t\t"<<"DNA
PATTERN DETECTION";
    cout<<"\n OPERATIONS
OFFERED";
    int k;
    cout<<"\n 1. To compare two DNAs
and know how much similar they are";
    cout<<"\n 2. To check how much
percentage of a given characteristics is
present in a DNA";
    cout<<"\n Choose an option to
proceed:";
```

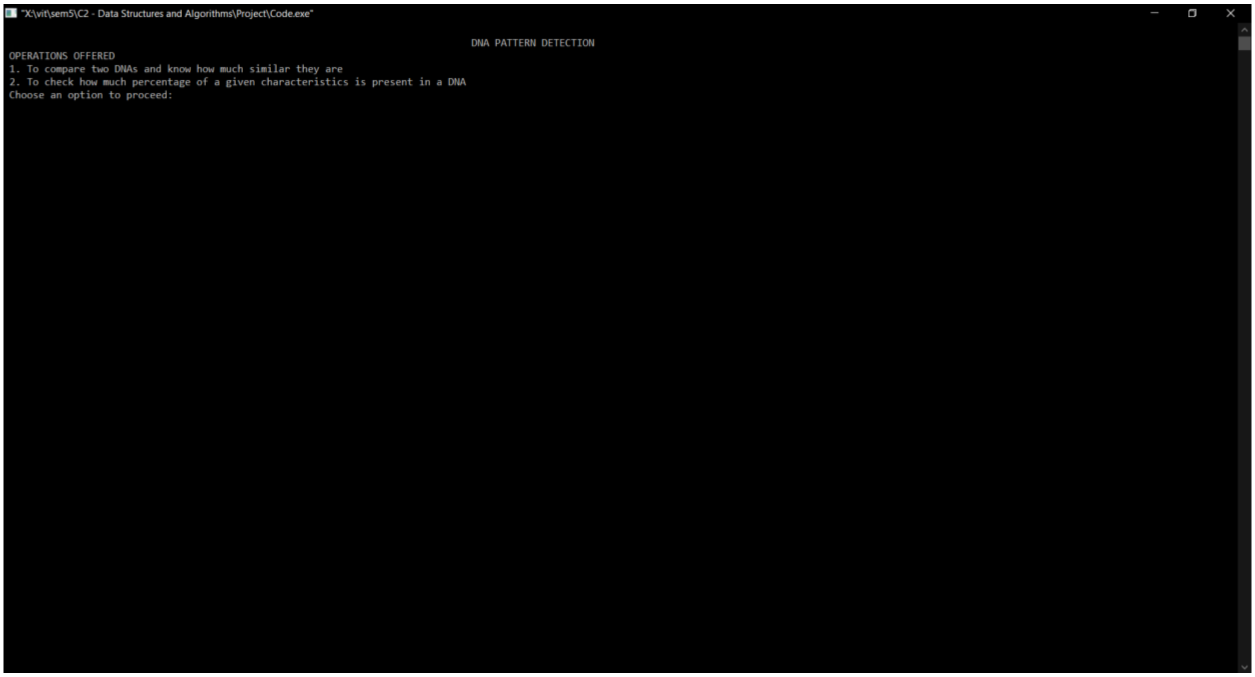
```
        }
        else if(D1S1[i]=='G'){
            if(D1S2[i]=='C'){
                a++;
            }
            else{
                b++;
                break;
            }
        }
        else if(D1S1[i]=='C'){
            if(D1S2[i]=='G'){
                a++;
            }
            else{
                b++;
                break;
            }
        }
        else{
            m++;
        }
    }
    if(b==0 && m==0){
        cout<<"\n The Inputed DNA is correct";
        return 1;
    }
    else{
        cout<<"\n The Inputed DNA is incorrect";
        return 0;
    }
}
int main(){
    welcomepg();
    getch();
    system("cls");
    return 0;
}
```

```
    cin>>k;
    switch(k){
        case 1:
            input1();
            break;
        case 2:
            input();
            break;
        default:
            cout<<"\n Invalid Input";
    }
}
```

6.4. Output:


"X:\vit\sem5\C2 - Data Structures and Algorithms\Project\Code.exe"

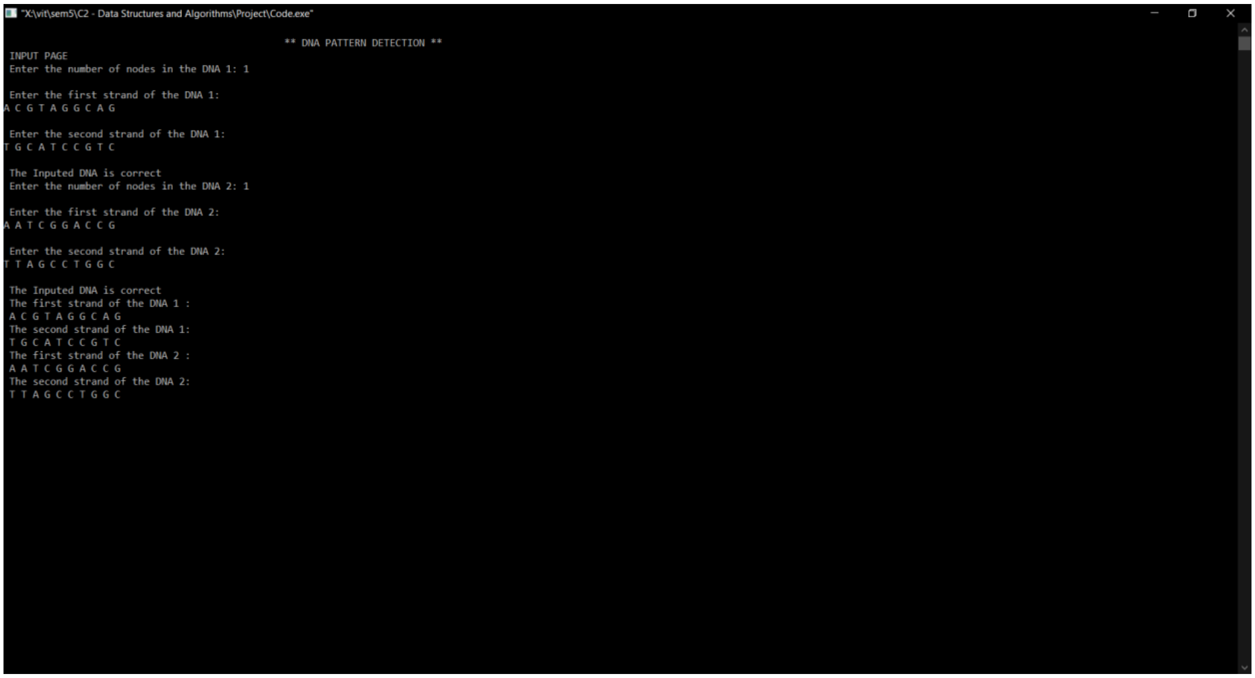
```
                                        DNA PATTERN DETECTION
1.About Us
2.Instructions
3.FAQ
4.Run
5.Exit
Choose an option to proceed:
```



"X:\vit\sem5\C2 - Data Structures and Algorithms\Project\Code.exe"

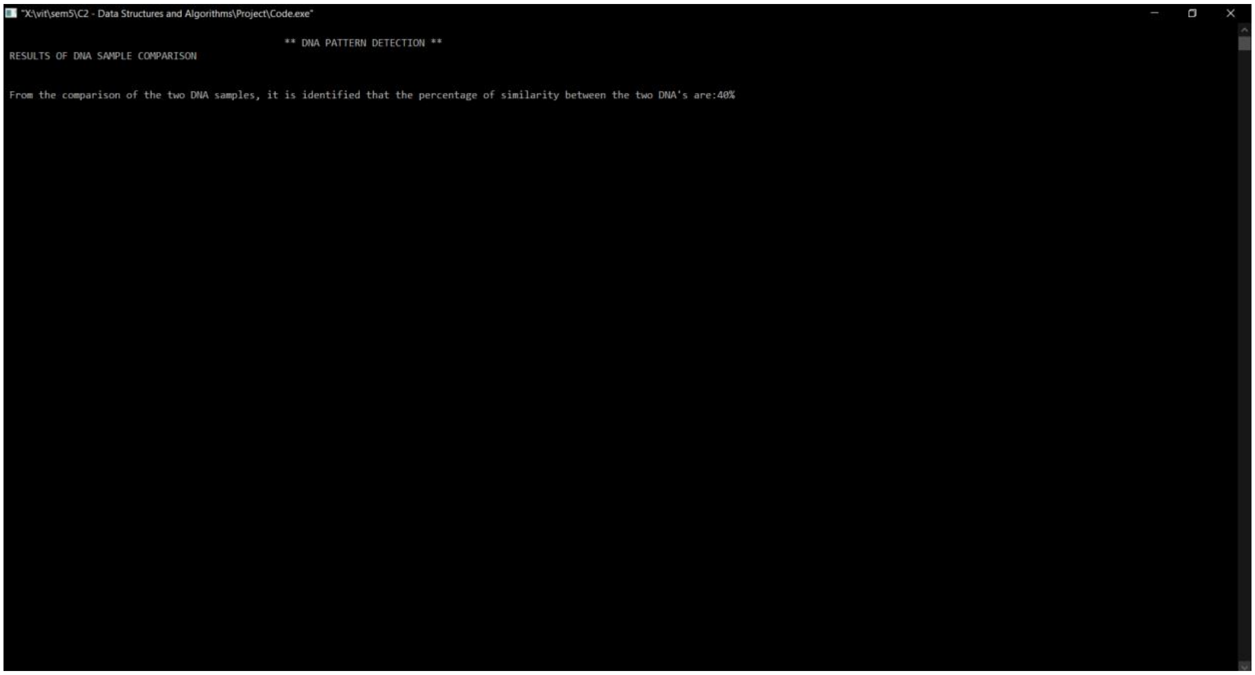
```
                                                                               DNA PATTERN DETECTION
OPERATIONS OFFERED
1. To compare two DNAs and know how much similar they are
2. To check how much percentage of a given characteristics is present in a DNA
Choose an option to proceed:
```

Comparing two DNA’s



"X:\vit\sem5\C2 - Data Structures and Algorithms\Project\Code.exe"

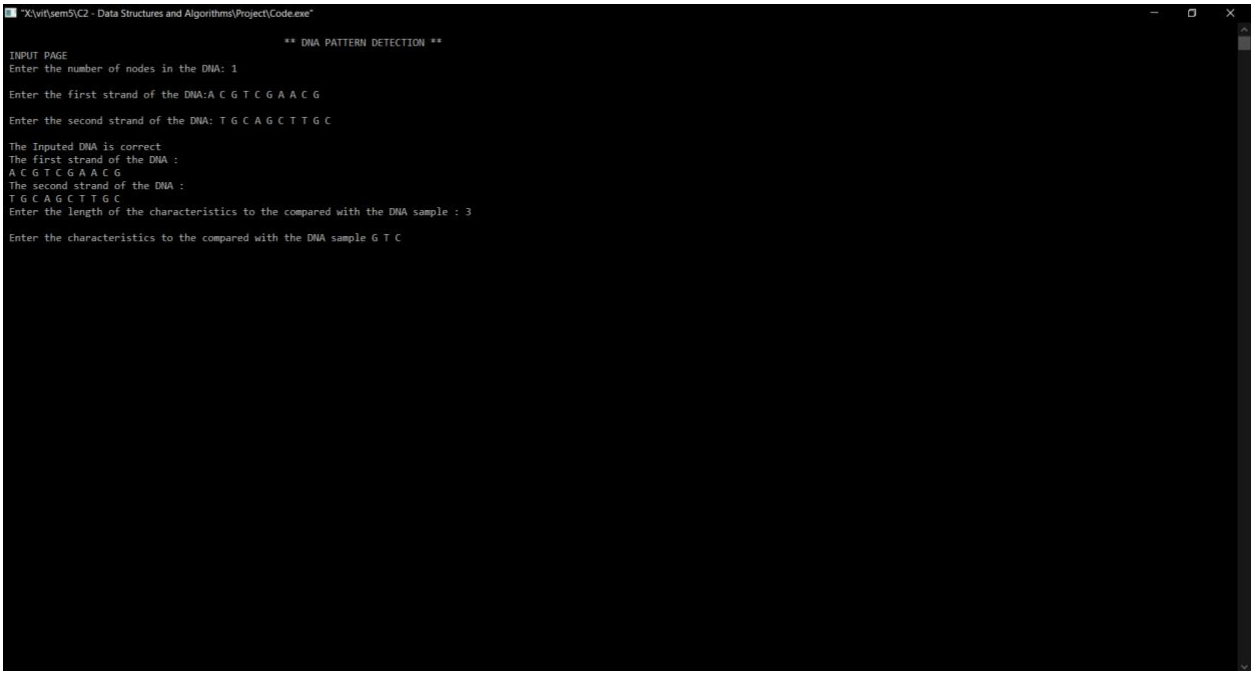
```
                                                ** DNA PATTERN DETECTION **
 INPUT PAGE
 Enter the number of nodes in the DNA 1: 1

 Enter the first strand of the DNA 1:
A C G T A G G C A G

 Enter the second strand of the DNA 1:
T G C A T C C G T C

 The Inputed DNA is correct
 Enter the number of nodes in the DNA 2: 1

 Enter the first strand of the DNA 2:
A A T C G G A C C G

 Enter the second strand of the DNA 2:
T T A G C C T G G C

 The Inputed DNA is correct
 The first strand of the DNA 1 :
 A C G T A G G C A G
 The second strand of the DNA 1:
 T G C A T C C G T C
 The first strand of the DNA 2 :
 A A T C G G A C C G
 The second strand of the DNA 2:
 T T A G C C T G G C
```



"X:\vit\sem5\C2 - Data Structures and Algorithms\Project\Code.exe"

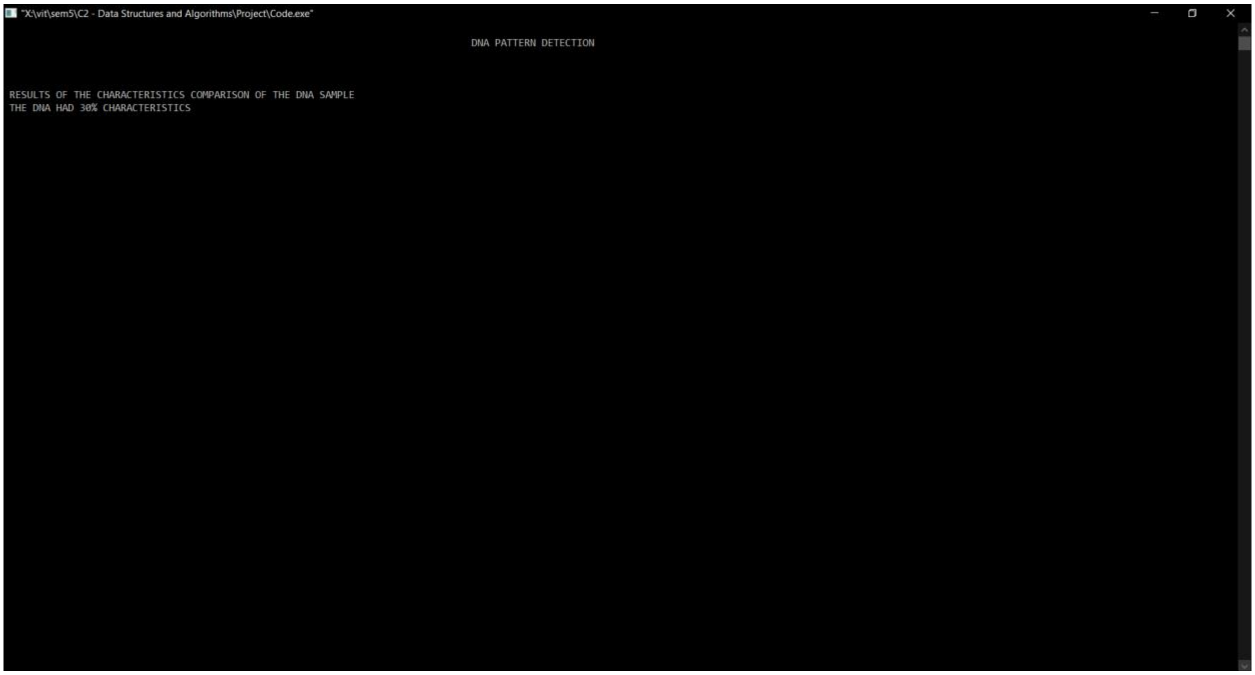
```
                                               ** DNA PATTERN DETECTION **
RESULTS OF DNA SAMPLE COMPARISON


From the comparison of the two DNA samples, it is identified that the percentage of similarity between the two DNA's are:40%
```

Finding the given characteristics in the DNA:



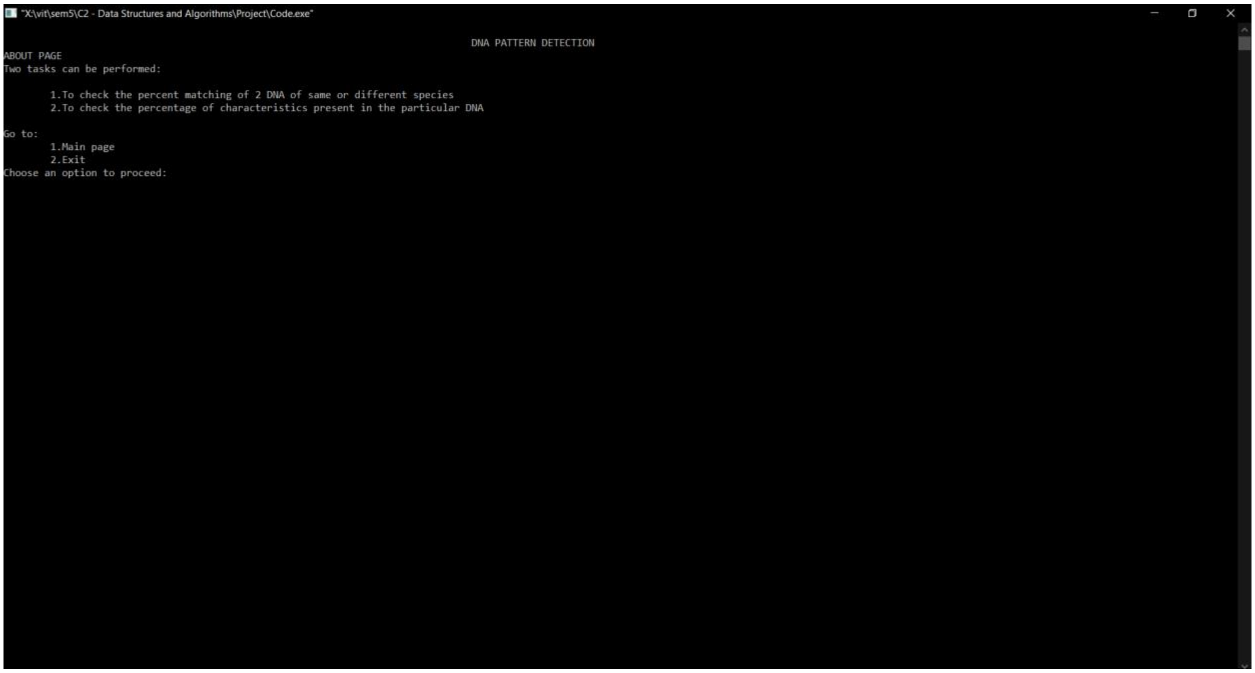
"X:\vit\sem5\C2 - Data Structures and Algorithms\Project\Code.exe"

```
                                               ** DNA PATTERN DETECTION **
INPUT PAGE
Enter the number of nodes in the DNA: 1

Enter the first strand of the DNA:A C G T C G A A C G

Enter the second strand of the DNA: T G C A G C T T G C

The Inputed DNA is correct
The first strand of the DNA :
A C G T C G A A C G
The second strand of the DNA :
T G C A G C T T G C
Enter the length of the characteristics to the compared with the DNA sample : 3

Enter the characteristics to the compared with the DNA sample G T C
```



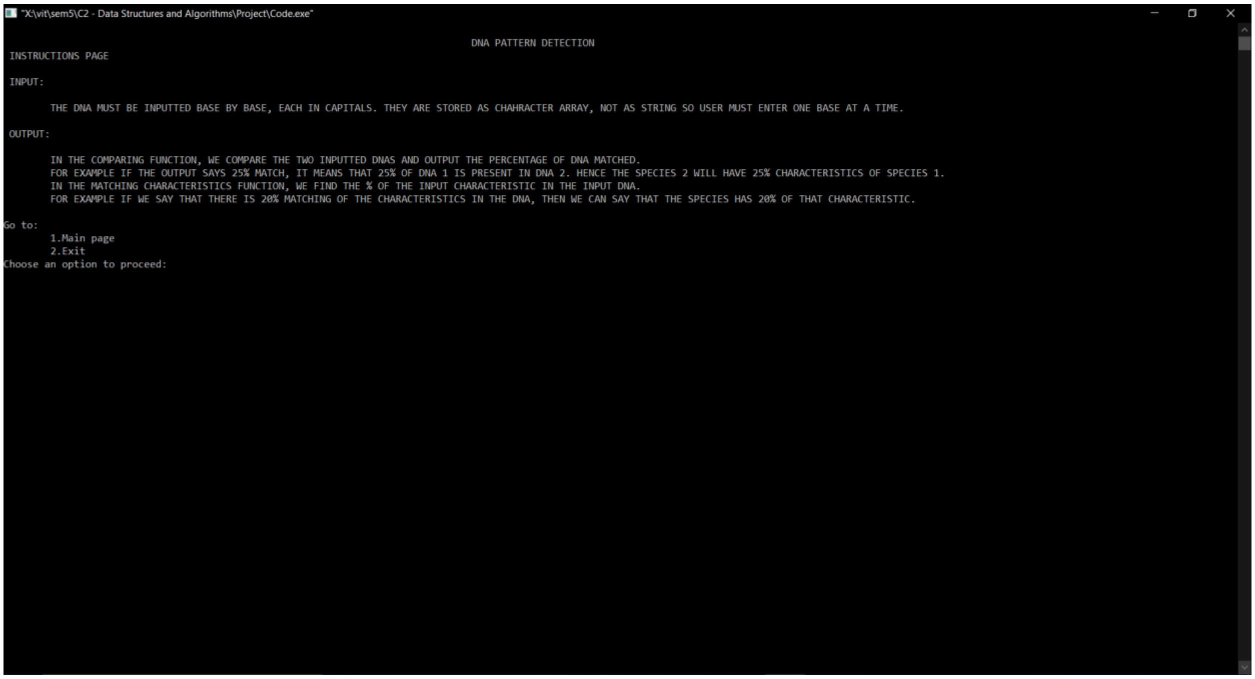
"X:\vit\sem5\C2 - Data Structures and Algorithms\Project\Code.exe"

```
                                                                                DNA PATTERN DETECTION



 RESULTS OF THE CHARACTERISTICS COMPARISON OF THE DNA SAMPLE
 THE DNA HAD 30% CHARACTERISTICS
```

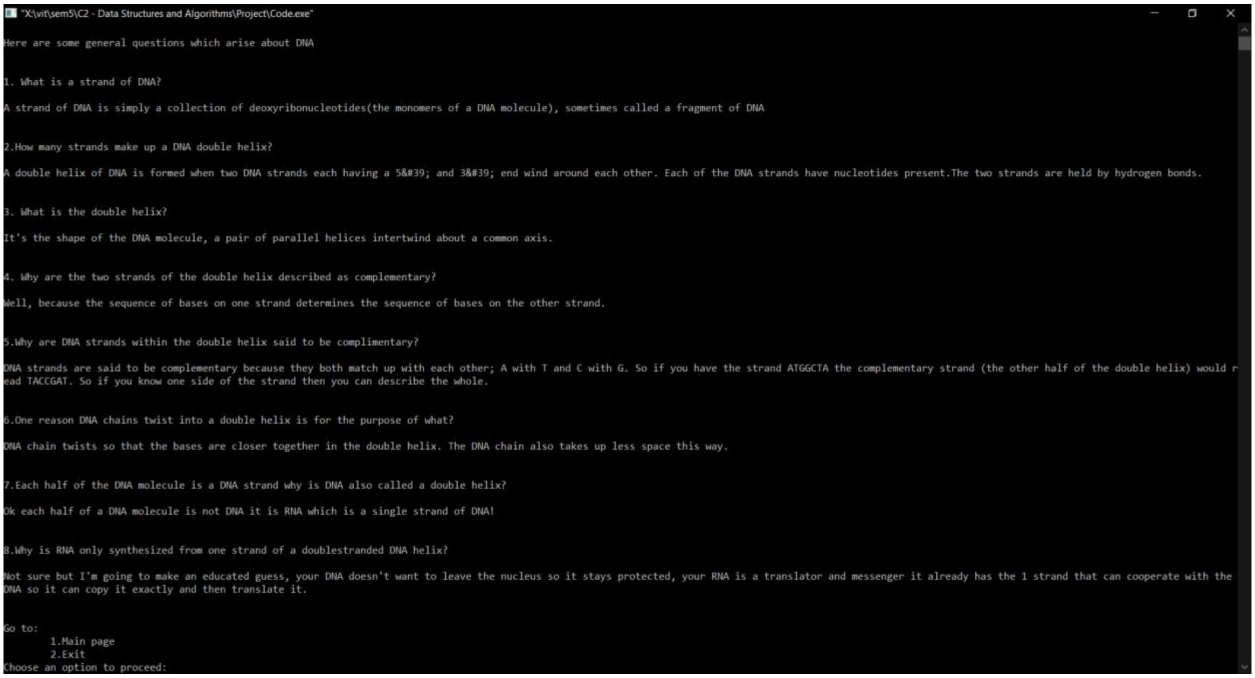


"X:\vit\sem5\C2 - Data Structures and Algorithms\Project\Code.exe"

```
                                                                                DNA PATTERN DETECTION
ABOUT PAGE
Two tasks can be performed:

        1.To check the percent matching of 2 DNA of same or different species
        2.To check the percentage of characteristics present in the particular DNA

Go to:
        1.Main page
        2.Exit
Choose an option to proceed:
```


"X:\vit\sem5\C2 - Data Structures and Algorithms\Project\Code.exe"

```
                                                                  DNA PATTERN DETECTION
 INSTRUCTIONS PAGE

 INPUT:

        THE DNA MUST BE INPUTTED BASE BY BASE, EACH IN CAPITALS. THEY ARE STORED AS CHAHRACTER ARRAY, NOT AS STRING SO USER MUST ENTER ONE BASE AT A TIME.

 OUTPUT:

        IN THE COMPARING FUNCTION, WE COMPARE THE TWO INPUTTED DNAS AND OUTPUT THE PERCENTAGE OF DNA MATCHED.
        FOR EXAMPLE IF THE OUTPUT SAYS 25% MATCH, IT MEANS THAT 25% OF DNA 1 IS PRESENT IN DNA 2. HENCE THE SPECIES 2 WILL HAVE 25% CHARACTERISTICS OF SPECIES 1.
        IN THE MATCHING CHARACTERISTICS FUNCTION, WE FIND THE % OF THE INPUT CHARACTERISTIC IN THE INPUT DNA.
        FOR EXAMPLE IF WE SAY THAT THERE IS 20% MATCHING OF THE CHARACTERISTICS IN THE DNA, THEN WE CAN SAY THAT THE SPECIES HAS 20% OF THAT CHARACTERISTIC.

Go to:
        1.Main page
        2.Exit
Choose an option to proceed:
```


"X:\vit\sem5\C2 - Data Structures and Algorithms\Project\Code.exe"

```
Here are some general questions which arise about DNA


1. What is a strand of DNA?

A strand of DNA is simply a collection of deoxyribonucleotides(the monomers of a DNA molecule), sometimes called a fragment of DNA


2.How many strands make up a DNA double helix?

A double helix of DNA is formed when two DNA strands each having a 5&#39; and 3&#39; end wind around each other. Each of the DNA strands have nucleotides present.The two strands are held by hydrogen bonds.


3. What is the double helix?

It's the shape of the DNA molecule, a pair of parallel helices intertwind about a common axis.


4. Why are the two strands of the double helix described as complementary?

Well, because the sequence of bases on one strand determines the sequence of bases on the other strand.


5.Why are DNA strands within the double helix said to be complimentary?

DNA strands are said to be complementary because they both match up with each other; A with T and C with G. So if you have the strand ATGGCTA the complementary strand (the other half of the double helix) would r
ead TACCGAT. So if you know one side of the strand then you can describe the whole.


6.One reason DNA chains twist into a double helix is for the purpose of what?

DNA chain twists so that the bases are closer together in the double helix. The DNA chain also takes up less space this way.


7.Each half of the DNA molecule is a DNA strand why is DNA also called a double helix?

Ok each half of a DNA molecule is not DNA it is RNA which is a single strand of DNA!


8.Why is RNA only synthesized from one strand of a doublestranded DNA helix?

Not sure but I'm going to make an educated guess, your DNA doesn't want to leave the nucleus so it stays protected, your RNA is a translator and messenger it already has the 1 strand that can cooperate with the
DNA so it can copy it exactly and then translate it.


Go to:
        1.Main page
        2.Exit
Choose an option to proceed:
```

7. Conclusion and Future Work

7.1 Conclusion:

Based on the results obtained, the KMP algorithm is able to detect the characteristics of the DNA very efficiently and it is very simple to implement it. Medical department can now use this DNA pattern detection using the KMP algorithm, they can easily find out the traits from the parent's DNA and also check if a particular characteristic is present in the given DNA and how intense is that characteristic.

7.2 Future Work

While experimenting, it was observed that the KMP algorithm is unable to indicate to what extend a match was found. Even then algorithm has nothing to do if a match is not found.

The use of partial match table may increase the space complexity of the KMP algorithm for relatively longer patterns. The time complexity in worst case is O(m + n). To hold the partial match table the algorithm requires O(m) more space. The future scope of this project is to develop a modified KMP algorithm, which has lesser space complexity.

8. References:

1. M. R. Hossen, A. M. Shafiul, and H. K. Rana, "Performance evaluation of various DNA pattern matching algorithms using different genome datasets," vol. 3, no. 1, pp. 14–18, 2018.

2. S. Rajesh, S. Prathima and D. Reddy, "Unusual Pattern Detection in DNA Database Using KMP Algorithm", *International Journal of Computer Applications*, vol. 1, no. 22, pp. 1-7, 2010. Available: 10.5120/526-687.

3. N. Kalita, Chitra, R. Sharma, and S. Borah, "EKMP: A proposed enhancement of KMP algorithm," in *Computational Intelligence in Data Mining - Volume 3*, New Delhi: Springer India, 2015, pp. 479–487.

4. P. Rahate and M. Chandak, "Comparative Study of String Matching Algorithms for DNA dataset", *International Journal of Computer Sciences and Engineering*, vol. 6, no. 5, pp. 1067-1074, 2018.

5. Alsmadi and M. Nuser, "String Matching Evaluation Methods for DNA Comparison", *International Journal of Advanced Science and Technology*, vol. 47, pp. 13-32,

6. T. M. Inbamalar and R. Sivakumar, "Improved algorithm for analysis of DNA sequences using multiresolution transformation," *ScientificWorldJournal*, vol. 2015, p. 786497, 2015

7. M YAZID M SAMAN, M NORDIN A RAHMAN, AZIZ AHMAD and A OSMAN M TAP, "A Minimum Cost Process in Searching for a Set of Similar DNA Sequences", *WSEAS*, pp. 348-353, 2006.

8. R. M. Zink and G. F. Barrowclough, "Mitochondrial DNA under siege in avian phylogeography: NuDNA VS. mtDNA in phylogeography," *Mol. Ecol.*, vol. 17, no. 9, pp. 2107–2121, 2008.

9. S. A. Marhon and S. C. Kremer, "Gene prediction based on DNA spectral analysis: a literature review," *J. Comput. Biol.*, vol. 18, no. 4, pp. 639–676, 2011.

10. P. Pandiyarajan, M. T and L. Raj, "A Comparative Study on String Matching Algorithm of Biological Sequences", *arXiv*, pp. 1-5, 2014.

11. S. Borah, D. Bhattacharjee, K. Kr. Singh and B. Rai, "Multithreaded KMP: A Proposed DNA Sequencing Algorithm", *International Journal of Advances in Science, Engineering and Technology(IJASEAT)*, vol. 3, no. 2,

12. I. Srinivas, M. Samnani and M. Shaikh, "Study of String Matching Algorithm", *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 1, no. 4, pp. 32-35, 2020.

13. 13. J. Lorenzo-Ginori, A. Rodriguez-Fuentes, R. Abalo, and R. Rodriguez, "Digital signal processing in the analysis of genomic sequences," *Curr. Bioinform.*, vol. 4, no. 1, pp. 28–40, 2009.

14. H. Ohmori, J. I. Tomizawa, and A. M. Maxam, "Detection of 5-methylcytosine in DNA sequences," *Nucleic Acids Res.*, vol. 5, no. 5, pp. 1479–1485, 1978.

15. D. Posada and K. A. Crandall, "Evaluation of methods for detecting recombination from DNA sequences: computer simulations," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 98, no. 24, pp. 13757–13762, 2001.