DNA AND HYBRIDIZATION ANALYZER

PROJECT REPORT

Submitted by

NAME	**REG.NO.**
Lakshay Nanda	16BCE0805
Keshika Tank	16BCE0796
Divyue Sharma	16BCE2011

Course Code: CSE2002

Course Title: THEORY OF COMPUTATION AND COMPILER DESIGN

Under the guidance of

Prof. Santhi K

Professor SCOPE

VIT University, Vellore.

Department of SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

NOVEMBER 2017

INDEX

1. ABSTRACT

In this project we are going to analyze the DNA sequence pattern entered by user and check it whether the give DNA sequence is valid or not. For this purpose the DNA pattern entered should belong to the specific language accepted by the analyzer which will be defined by us. Also we are going to give option to user to enter more than one DNA sequence and then we will check there validity and also whether the hybridization between the two DNA entered is possible or not. Hence it would form a lexical analyzer along with the computation from the given strings of the language. So with this we can easily check whether the DNA strands are properly entered with respect to the language and whether we can hybridize them to form newly hybridized DNAs by checking some similarity conditions for each pair.

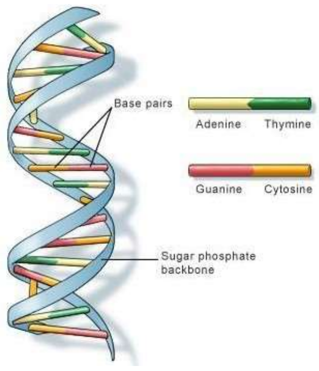
2. INTRODUCTION

In this project we are going to build a lexical analyzer which will scan the DNA protein sequence and would also help us to check the validity of the sequence. Along with this we would make another hybridization parser which would show us the results whether hybridization between two DNA strands is possible or not and if possible what would be the net resultant strand of the DNA formed by the hybridization.

About DNA

DNA or otherwise called deoxyribonucleic acid is the building block of the life. It contains the information the cell requires to synthesize protein and to replicate itself, to be short it is the storage repository for the information that is required for any cell to function. The properties of DNA is that a protein can combine with only one as its valid counterpart. Hence only a specific combination is valid for a DNA strand which needs to be continued along the length of the strand.

DNA, or deoxyribonucleic acid, is the hereditary material in humans and almost all other organisms. Nearly every cell in a person's body has the same DNA. The information in DNA is stored as a code made up of four chemical bases: adenine (A), guanine (G), cytosine (C), and thymine (T). Human DNA consists of about 3 billion bases, and more than 99 percent of those bases are the same in all people. The order, or sequence, of these bases determines the information available for building and maintaining an organism, similar to the way in which letters of the alphabet appear in a certain order to form words and sentences.



Seeing the pairs, we have defined our own language for the sequence of DNA and designed a grammar suitable grammar for the above purpose.

About Hybridization

Hybridization is the process of combining two complementary single-stranded DNA or RNA molecules and allowing them to form a single double-stranded molecule through base pairing.

DNA hybridization measures the degree of genetic similarity between complete genomes by measuring the amount of heat required to melt the hydrogen bonds between the base pairs that form the links between the two strands of the double helix of duplex DNA. The comparison may be between the two DNA strands of an individual or of different individuals representing different levels of genetic divergence.

3. BACKGROUND OF THE WORK

About lexical analyzer

Lexical analyzer is the first phase of a compiler. It takes the modified source code from language preprocessors that are written in the form of sentences. The lexical analyzer breaks these syntaxes into a series of tokens, by removing any whitespace or comments in the source code which in our case would be the each protein separated into a token. It verifies the grammar rules defined by us and accordingly returns whether the DNA entered by the user is correct or not.

About hybridization evaluator

From this evaluator, we can check whether the pair of DNA can be hybridized. For this, we would check whether there are some same DNA pairs which are essential for hybridization. If the condition satisfy then we can hybridize them in 2 ways:

(i) FIRST PART (DNA 1) , MATCHING PART (DNA 1 & 2), SECOND PART (DNA 2)

(ii) FIRST PART (DNA 2) , MATCHING PART (DNA 1 & 2), SECOND PART (DNA 1)

4. OVERVIEW OF THE PROJECT

In this project we are going to analyze the DNA sequence entered by the user and after checking the validity considering all the properties of a DNA, we are also going to show that whether the two or more DNA sequences entered by the user can be hybridized or not. And if hybridization is possible then what will be the output DNA strand. For this purpose we define our own grammar for the input DNA sequences we will provide to check firstly its validity and then checking the Hybridization condition for a pair of input of DNA sequence.

Problem description

In daily basis life it is quiet hard for anyone to check whether the given DNA sample is valid or not and whether the Hybridization between the given DNA sequences is possible or not and if possible then what will be the hybridized result. And so we made a program which can solve this problem quiet easily.

Working model

Grammar

The Language Followed:

Alphabets: {A, a, T, t, G, g, C, c,$} Grammar rules:

- After pair of 2 alphabets ‘,’ is needed.
- For a pair starting with A or a must have T or t as next alphabet.
- For a pair starting with G or g must have C or c as next alphabet.
- For a pair starting with C or c must have G or g as next alphabet.
- For a pair starting with T or t must have A or a as next alphabet.
- In this the ‘,’ is used to check the next pattern of DNA. For example if ‘AT,GC,CG,TA’ is entered then Analyzer will first read ‘AT’ then ‘,’ will inform it that this pattern or word is over then analyzer will read next pattern that in this example is ‘GC’.

Some accepting strings

At,gc$

GC,CG$ tA

cG,AT$

gc,GC,AT$

Some non-accepting strings

A

GC,

,AT GT CA Atg

Epsilon€

,

Let grammar be defined by G(V, £, P, S)

Here,

V={‘$’,a,A,t,T,g,G,c,C,’,’}

£={ $ }

S={ a,A,t,T,g,G,c,C } Productions

(p):

S -> X '$' | X ',' Y '$'

X -> P | Q | R |O

Y -> X ',' Y | X

P -> 'a' 't' | 'A' 't' | 'A' 'T' | 'a' 'T'

Q -> 't' 'a' | 'T' 'a' | 'T' 'A' | 't' 'A'

R -> 'c' 'g' | 'C' 'g’ | 'C' 'G' | 'c' 'G'

O -> 'g' 'c' | 'G' 'c' | 'G' 'C' | 'g' '

Design description

We have designed an interactive C executable file which include the name of application as it starts and then goes to the next stage which is the execution stage. Here we get a detailed analysis about the DNA sequences and the hybridizations possible of the two DNA sequences. The executable file is properly designed with background colour and font colour to make it look more attractive.

5. IMPLEMENTATION

Description of Modules/Programs

- **LOADING**

The loading function improves the user interface of the executing application file by showing the start page first and then allowing user to enter the input belonging to the language of DNA described by our grammar. It bypasses them to the real application and acts as a start screen.

- **SUBSTRING**

This function performs a task of breaking the string into the substring where we need to give the input as start index, end index and the string and the result is the substring of that particular string.

- **LEXICAL_ANALYSER**

The function lexical_analyser checks the validity of the input string according to the grammar rules of our grammar taking into consideration the productions of our grammar. Firstly, the function checks whether the input string length is zero, i.e., no input, or not. We check the input string character by character and check whether that character is paired with its appropriate counterpart (A with T or G with C and vice versa). Also it checks whether there's a comma between every pair of the two DNA strands. Along with this, it also checks whether the end marker is present at the end of each input string or not. It exits, if any of the grammar rules is not obeyed otherwise returns that the input is lexically and grammatically correct.

- **HYBRIDIZATION**

Hybridization function processes two input DNA sequences and check whether the length of both the inputs is same or not and the basic hybridization condition which is minimum three pairs should be same and the prints the final hybridized DNA sequences. It outputs both the possible hybridized sequences, first is starting set of input1 then the common part followed by the ending of input2 and the second is starting set of input2 then the common part followed by the ending of input1.

Source Code

```
#include <stdio.h>

#include <string.h>

#include <stdlib.h>


char final1[100], final2[100],
     sub_string[100];
void substring(char string[], char
     sub[], int start, int end) {

  int c = start;

  int i = 0;
  while (c

    <= end)

   {
    sub[i] =

      string[c];
    i++;
    c++;

   }
  sub[c] = '\0';
  strcpy(sub_string, sub);

}
void lexical_analyser(char input[200])

{
  if (strlen(input) == 0)

   {
    printf("No input entered");
  exit(-1);

}
int j = 0;
input = strupr(input);
char i = input[j];
while (i != NULL) {

  printf("%c --> ", i);
  switch (i)

  {

  case 'A':

    printf("Protein Adenine\n");

    if (input[j + 1] == 'T' && ((input[j
   + 2] == ',') ||

        (input[j + 2] == '$')))

    {

      printf("%c --> ", input[j + 1]);

      printf("Protein Thyamine\n");

      printf("Correct pair\n");

      if (input[j + 2] == ',')

      {

        printf("%c --> ", input[j + 2]);

        printf("One set over. Waiting
   for the other set\n");

        if (input[j + 3] == NULL)

        {

          printf("Not the correct way to
end the string. Does not belong to
the language");
       exit(-1);

     }

   }

 } else

 {

   if (input[j + 1] == NULL)

   {

     printf("Protein cannot exist
without its other complementary
part. ");

   } else

   {

     printf("Wrong combination as
A can make pair with T only.\n");

   }

   printf("Does not belong to the
language.\n");

   exit(-1);

 }

 j = j + 2;

 break;

case 'T':

 printf("Protein Thyamine\n");
 if (input[j + 1] == 'A' && ((input[j
+ 2] == ',') || (input[j + 2] == '$')))
```

```
 {

   printf("%c --> ", input[j + 1]);

   printf("Protein Adenine\n");

   printf("Correct pair\n");

   if (input[j + 2] == ',')

   {

     printf("%c --> ", input[j + 2]);

     printf("One set over. Waiting
for the other set\n");

     if (input[j + 3] == NULL)

     {

       printf("Not the correct way to
end the string. Does not belong to
the language");
       exit(-1);

     }

   }

 } else

 {

   if (input[j + 1] == NULL)

   {

     printf("Protein cannot exist
without its other complementary
part. ");

   } else
```

```
    {

      printf("Wrong combination as T
  can make pair with A only.\n");

    }

    printf("Does not belong to the
  language.\n");

    exit(-1);

  }

  j = j + 2;

  break;

case 'G':

  printf("Protein Guanine\n");
  if (input[j + 1] == 'C' && ((input[j
  + 2] == ',') || (input[j + 2] == '$')))

  {

    printf("%c --> ", input[j + 1]);

    printf("Protein Cytosine\n");

    printf("Correct pair\n");

    if (input[j + 2] == ',')

    {

      printf("%c --> ", input[j + 2]);

      printf("One set over. Waiting
  for the other set\n");

      if (input[j + 3] == NULL)

      {
        printf("Not the correct way to
  end the string. Does not belong to
  the language");
        exit(-1);

      }

    }

  } else

  {

    if (input[j + 1] == NULL)

    {

      printf("Protein cannot exist
  without its other complementary
  part. ");

    } else

    {

      printf("Wrong combination as
  G can make pair with C only.\n");

    }

    printf("Does not belong to the
  language.\n");

    exit(-1);

  }
  j = j + 2;
  break;
case 'C':

  printf("Protein Cytosine\n");

  if (input[j + 1] == 'G' && ((input[j
  + 2] == ',') || (input[j + 2] == '$')))
```

```
{

  printf("%c --> ", input[j + 1]);

  printf("Protein Guanine\n");

  printf("Correct pair\n");

  if (input[j + 2] == ',')

  {

    printf("%c --> ", input[j + 2]);

    printf("One set over. Waiting
for the other set\n");

    if (input[j + 3] == NULL)

    {

      printf("Not the correct way to
end the string. Does not belong to
the language");
      exit(-1);

    }

  }

} else

{

  if (input[j + 1] == NULL)

  {

    printf("Protein cannot exist
without its other complementary
part. ");

  } else
```

```
   {

     printf("Wrong combination as
 C can make pair with G only.\n");

   }

   printf("Does not belong to the
 language.\n");

   exit(-1);

  }

  j = j + 2;

  break;

case '$':

  if (i == '$' && input[strlen(input) -
 1] == '$' && input[j + 1] ==
 NULL)

  {

   printf("String termination\n");

  } else

  {

   printf("Inappropriate use of
 '$'\n");

   exit(-1);

  }

  break;

default:

  printf("Not a valid character. Does
 not belong to the alphabet set of
```

```
    the language. Hence does not
    belong to the language.\n");

     exit(-1);

    }
    i = input[++j];

   }

  printf("\nTHE INPUT ENTERED IS
    LEICALLY CORRECT\n\n");

 }
void hybridization(char input1[200],
    char input2[200])

{
  int k = 0, l = 0, s = 0;
  printf("\nEnter no. of pairs to check
    the suitability for
    hybridization:\n");

  scanf("%d", & s);
  char temp[100];
  int same = 0, flag = 0;
  char sub1[100], sub2[100];
  if (strlen(input1) == strlen(input2)) {
    for (k = 0; k + 1 + (s -1) * 3 <
    strlen(input1); k = k + 3)

    {
     for (l = 0; l + 1 + (s - 1) * 3 <
    strlen(input2); l = l + 3)

     {

      substring(input1, sub1, k, k + 1 +
    (s - 1) * 3);
      strcpy(sub1, sub_string);
      substring(input2, sub2, l, l + 1 +
    (s - 1) * 3);
      strcpy(sub2, sub_string);

      if (strcmpi(sub1, sub2) == 0)
  {

    same = 1;

    printf("Hybridization possible
as at least %d pairs are same\n", s);
    if (k != 0)

    {

      substring(input1, temp, 0, k -
1);

      strcat(final1, sub_string);

    }

    strcat(final1, sub1);

    substring(input2, temp, l + 2 +
(s - 1) * 3, strlen(input2));

    strcat(final1, sub_string);

    if (l != 0)

    {

      substring(input2, temp, 0, l -
1);

      strcat(final2, sub_string);

    }

    strcat(final2, sub2);

    substring(input1, temp, k + 2 +
(s - 1) * 3, strlen(input1));

    strcat(final2, sub_string);

    printf("1st Hybridization result-
----------%s\n", final1);
```

```
          printf("2nd Hybridization
      result-----------%s\n", final2);

          flag = 1;

          break;

        }

      }

      if (flag == 1)

      {

        break;

      }

    }
    if (same == 0) {
      printf("Hybridization not possible
      as at least %d pairs are not
      same\n", s);

      exit(-1);

    }

  } else {
    printf("DNA strands are not of
     same size. Hence hybridization not
     possible");
    exit(1);

  }
}
void main() {
  int i = 0;

  char a1[200], b1[200];

  char line[100], a[100], b[100],
     temp[100];
  FILE * input = fopen("text.txt", "r");
```

```
while (fgets(line, 25, input) !=
   NULL)

{

  if (i == 0)

  {
    strcpy(a, line);

    i++;

  } else

  {

    strcpy(b, line);

  }

}

printf("DNA 1---------------------------
   -------------------------------------\n");

printf("%s\n", a);
lexical_analyser(a);
printf("DNA 2---------------------------
   -------------------------------------\n");

printf("%s\n", b);

lexical_analyser(b);

printf("Checkingfor
   hybridization\n");

substring(a, temp, 0, strlen(a) - 2);

strcpy(a, sub_string);
printf("%s\n", a);

substring(b, temp, 0, strlen(b) - 2);
```

```
strcpy(b, sub_string);
printf("%s\n", b);

hybridization(a, b);
fclose(input);
```

 }**Text file**

aT,ta,GC,CG,gc,ta,at,cg$

gc,cg,GC,at,ta,gc,at,ta$

Test cases

1. aT,ta,GC,CG,gc,ta,at,cg$

gc,cg,GC,at,ta,gc,at,ta$

where n=3

n=2

and n=5

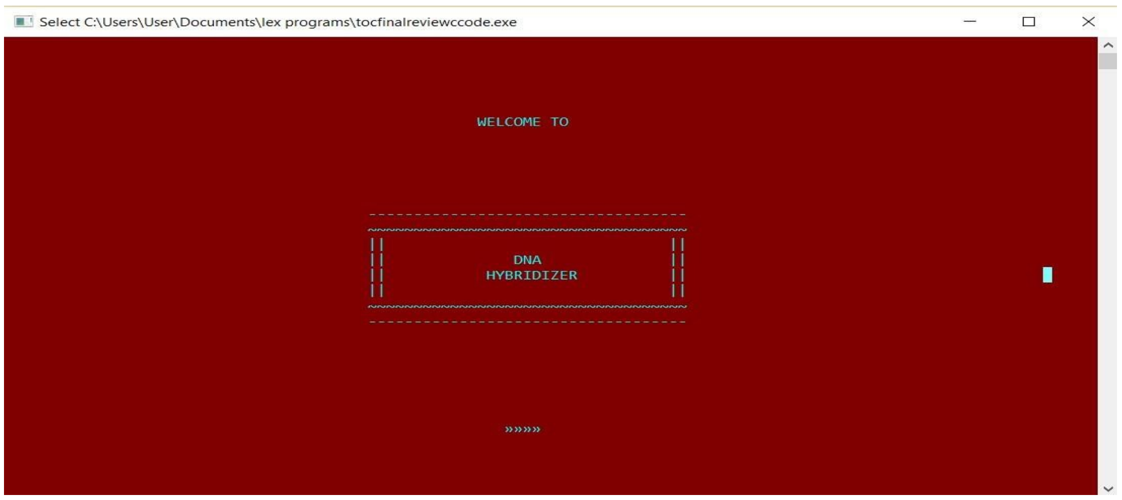
2. gc,CG,gc,cg,at,ta,at,cg$

   gc,cg,GC,at,ta,gc,at,ta$

   where n=3

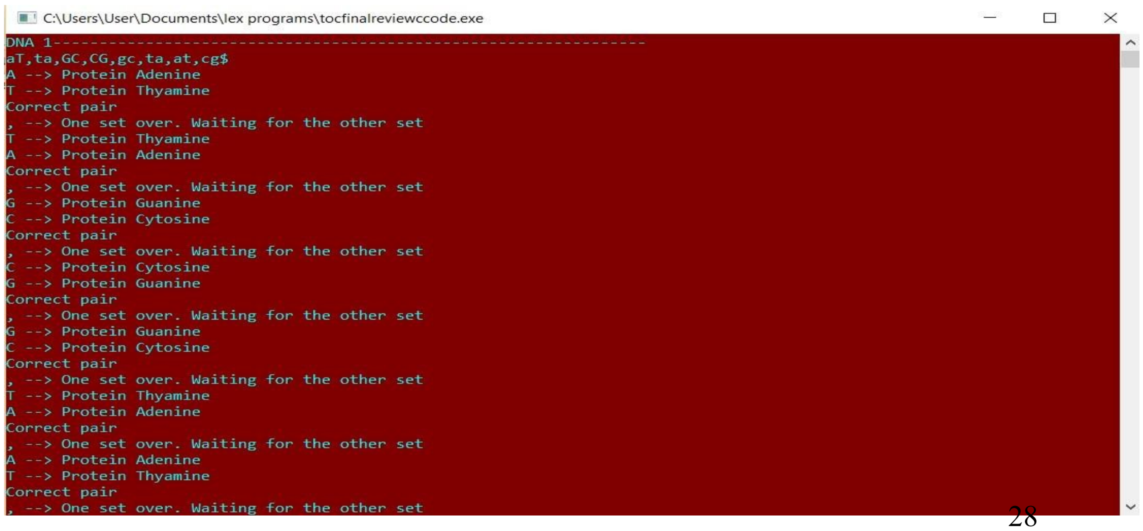
   n=4

Execution snapshots

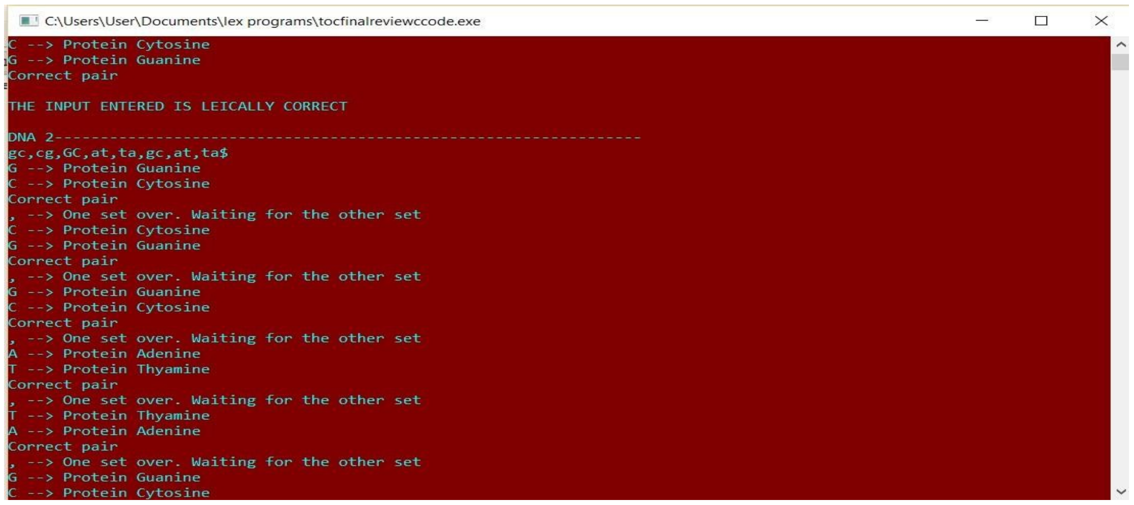


```
Select C:\Users\User\Documents\lex programs\tocfinalreviewccode.exe

                              WELCOME TO

                 ------------------------------------
                 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                 ||                                ||
                 ||             DNA                ||
                 ||          HYBRIDIZER            ||
                 ||                                ||
                 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                 ------------------------------------

                              »»»»»
```

1.

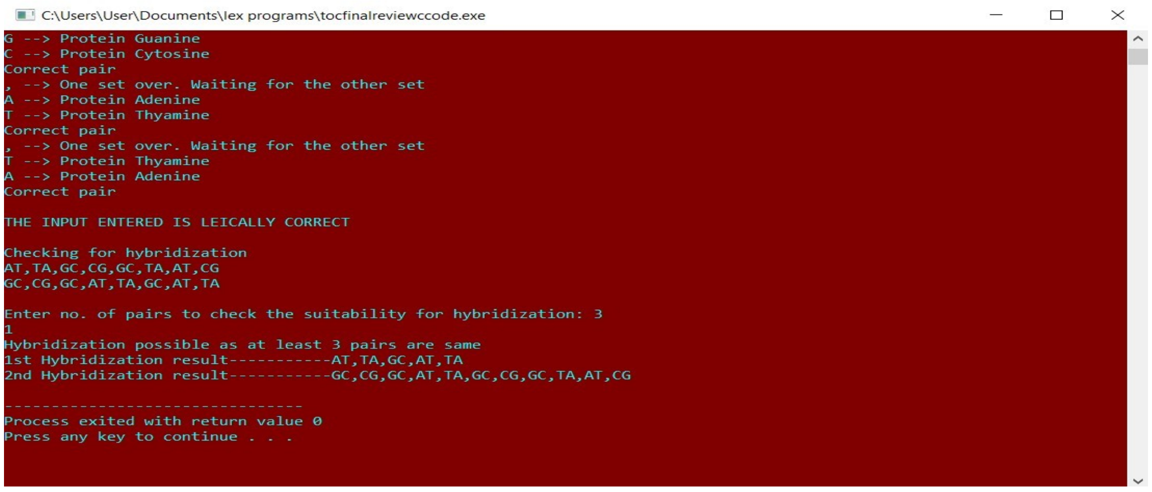


```
C:\Users\User\Documents\lex programs\tocfinalreviewccode.exe
DNA 1-------------------------------------------------------------
aT,ta,GC,CG,gc,ta,at,cg$
A --> Protein Adenine
T --> Protein Thyamine
Correct pair
, --> One set over. Waiting for the other set
T --> Protein Thyamine
A --> Protein Adenine
Correct pair
, --> One set over. Waiting for the other set
G --> Protein Guanine
C --> Protein Cytosine
Correct pair
, --> One set over. Waiting for the other set
C --> Protein Cytosine
G --> Protein Guanine
Correct pair
, --> One set over. Waiting for the other set
G --> Protein Guanine
C --> Protein Cytosine
Correct pair
, --> One set over. Waiting for the other set
T --> Protein Thyamine
A --> Protein Adenine
Correct pair
, --> One set over. Waiting for the other set
A --> Protein Adenine
T --> Protein Thyamine
Correct pair
, --> One set over. Waiting for the other set
```



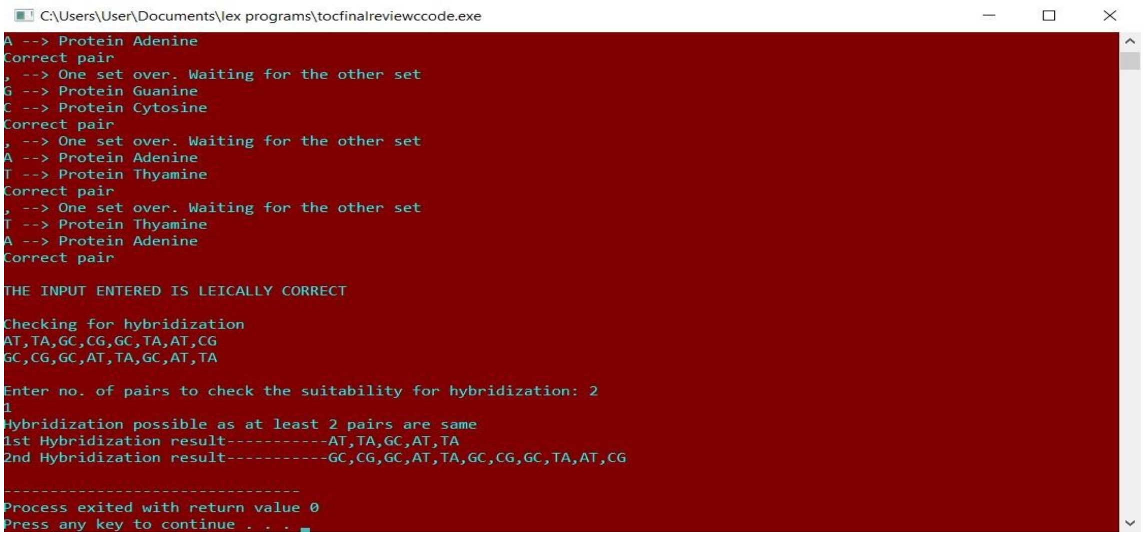
C:\Users\User\Documents\lex programs\tocfinalreviewccode.exe

```
C --> Protein Cytosine
G --> Protein Guanine
Correct pair

THE INPUT ENTERED IS LEICALLY CORRECT

DNA 2-----------------------------------------------------------------
gc,cg,GC,at,ta,gc,at,ta$
G --> Protein Guanine
C --> Protein Cytosine
Correct pair
, --> One set over. Waiting for the other set
C --> Protein Cytosine
G --> Protein Guanine
Correct pair
, --> One set over. Waiting for the other set
G --> Protein Guanine
C --> Protein Cytosine
Correct pair
, --> One set over. Waiting for the other set
A --> Protein Adenine
T --> Protein Thyamine
Correct pair
, --> One set over. Waiting for the other set
T --> Protein Thyamine
A --> Protein Adenine
Correct pair
, --> One set over. Waiting for the other set
G --> Protein Guanine
C --> Protein Cytosine
```



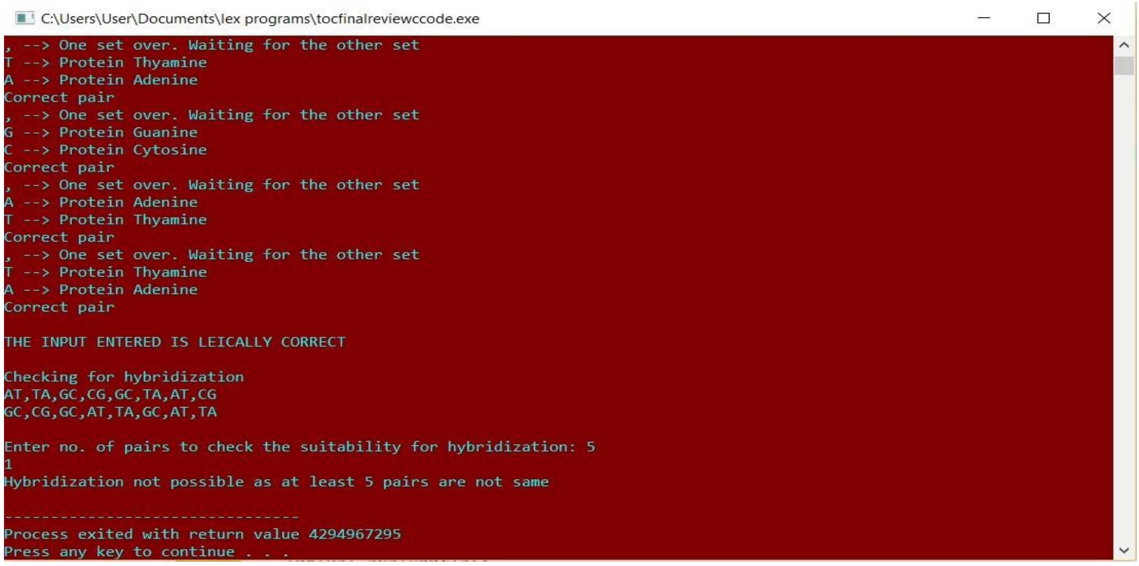
C:\Users\User\Documents\lex programs\tocfinalreviewccode.exe

```
G --> Protein Guanine
C --> Protein Cytosine
Correct pair
, --> One set over. Waiting for the other set
A --> Protein Adenine
T --> Protein Thyamine
Correct pair
, --> One set over. Waiting for the other set
T --> Protein Thyamine
A --> Protein Adenine
Correct pair

THE INPUT ENTERED IS LEICALLY CORRECT

Checking for hybridization
AT,TA,GC,CG,GC,TA,AT,CG
GC,CG,GC,AT,TA,GC,AT,TA

Enter no. of pairs to check the suitability for hybridization: 3
1
Hybridization possible as at least 3 pairs are same
1st Hybridization result-----------AT,TA,GC,AT,TA
2nd Hybridization result-----------GC,CG,GC,AT,TA,GC,CG,GC,TA,AT,CG

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```



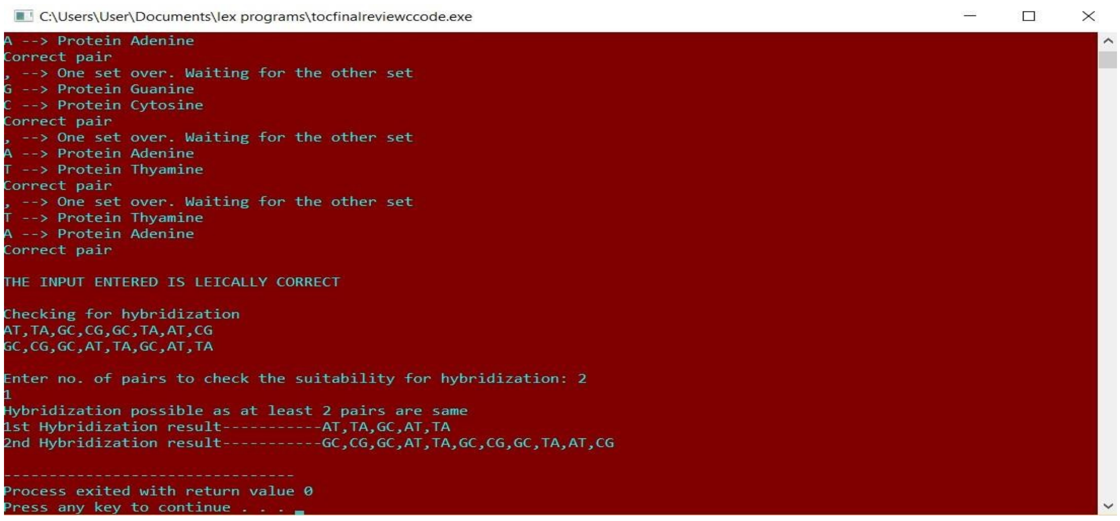
C:\Users\User\Documents\lex programs\tocfinalreviewccode.exe

```
A --> Protein Adenine
Correct pair
, --> One set over. Waiting for the other set
G --> Protein Guanine
C --> Protein Cytosine
Correct pair
, --> One set over. Waiting for the other set
A --> Protein Adenine
T --> Protein Thyamine
Correct pair
, --> One set over. Waiting for the other set
T --> Protein Thyamine
A --> Protein Adenine
Correct pair

THE INPUT ENTERED IS LEICALLY CORRECT

Checking for hybridization
AT,TA,GC,CG,GC,TA,AT,CG
GC,CG,GC,AT,TA,GC,AT,TA

Enter no. of pairs to check the suitability for hybridization: 2
1
Hybridization possible as at least 2 pairs are same
1st Hybridization result-----------AT,TA,GC,AT,TA
2nd Hybridization result-----------GC,CG,GC,AT,TA,GC,CG,GC,TA,AT,CG

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```



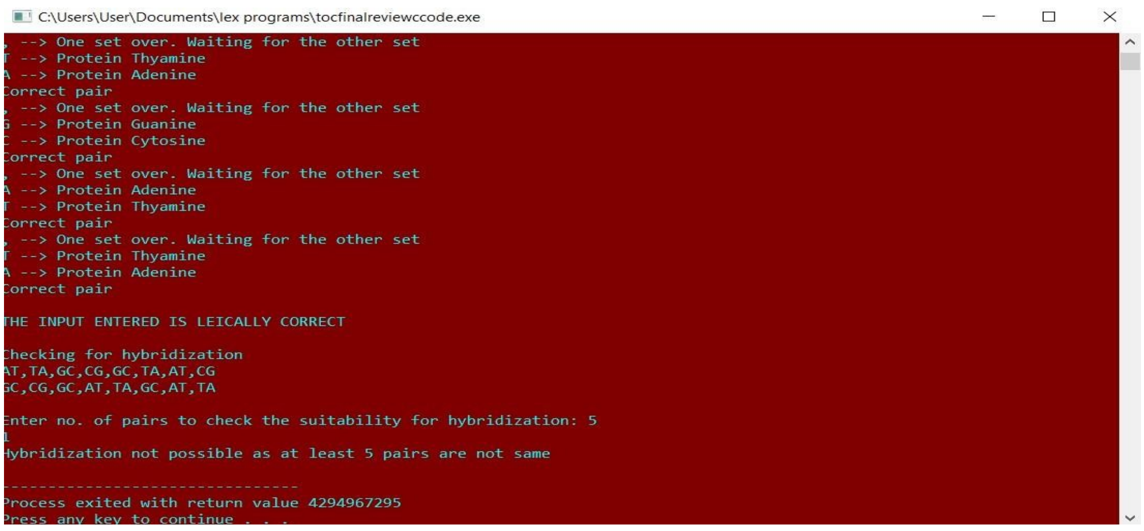
C:\Users\User\Documents\lex programs\tocfinalreviewccode.exe

```
, --> One set over. Waiting for the other set
T --> Protein Thyamine
A --> Protein Adenine
Correct pair
, --> One set over. Waiting for the other set
G --> Protein Guanine
C --> Protein Cytosine
Correct pair
, --> One set over. Waiting for the other set
A --> Protein Adenine
T --> Protein Thyamine
Correct pair
, --> One set over. Waiting for the other set
T --> Protein Thyamine
A --> Protein Adenine
Correct pair

THE INPUT ENTERED IS LEICALLY CORRECT

Checking for hybridization
AT,TA,GC,CG,GC,TA,AT,CG
GC,CG,GC,AT,TA,GC,AT,TA

Enter no. of pairs to check the suitability for hybridization: 5
1
Hybridization not possible as at least 5 pairs are not same

--------------------------------
Process exited with return value 4294967295
Press any key to continue . . .
```



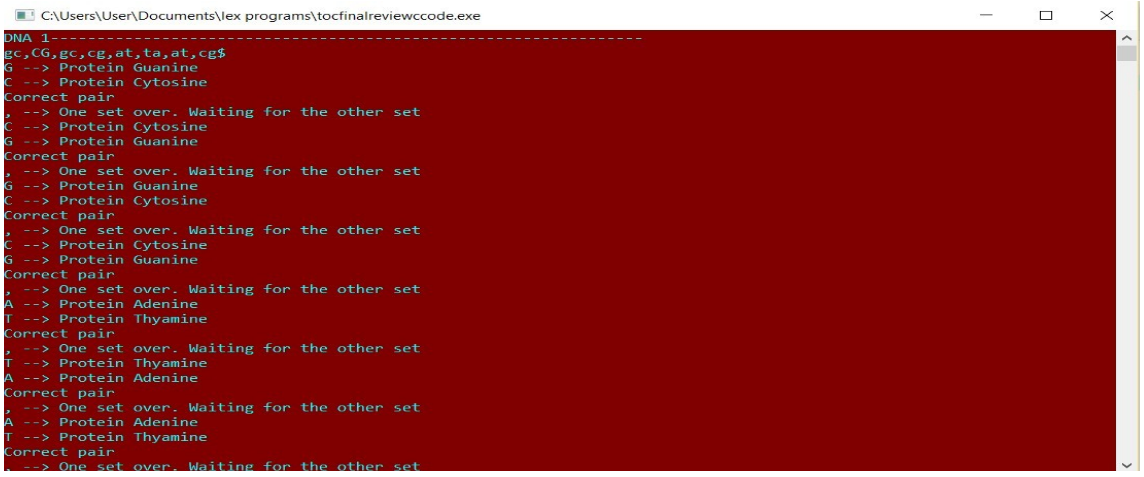
C:\Users\User\Documents\lex programs\tocfinalreviewccode.exe

```
A --> Protein Adenine
Correct pair
, --> One set over. Waiting for the other set
G --> Protein Guanine
C --> Protein Cytosine
Correct pair
, --> One set over. Waiting for the other set
A --> Protein Adenine
T --> Protein Thyamine
Correct pair
, --> One set over. Waiting for the other set
T --> Protein Thyamine
A --> Protein Adenine
Correct pair

THE INPUT ENTERED IS LEICALLY CORRECT

Checking for hybridization
AT,TA,GC,CG,GC,TA,AT,CG
GC,CG,GC,AT,TA,GC,AT,TA

Enter no. of pairs to check the suitability for hybridization: 2
1
Hybridization possible as at least 2 pairs are same
1st Hybridization result-----------AT,TA,GC,AT,TA
2nd Hybridization result-----------GC,CG,GC,AT,TA,GC,CG,GC,TA,AT,CG

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

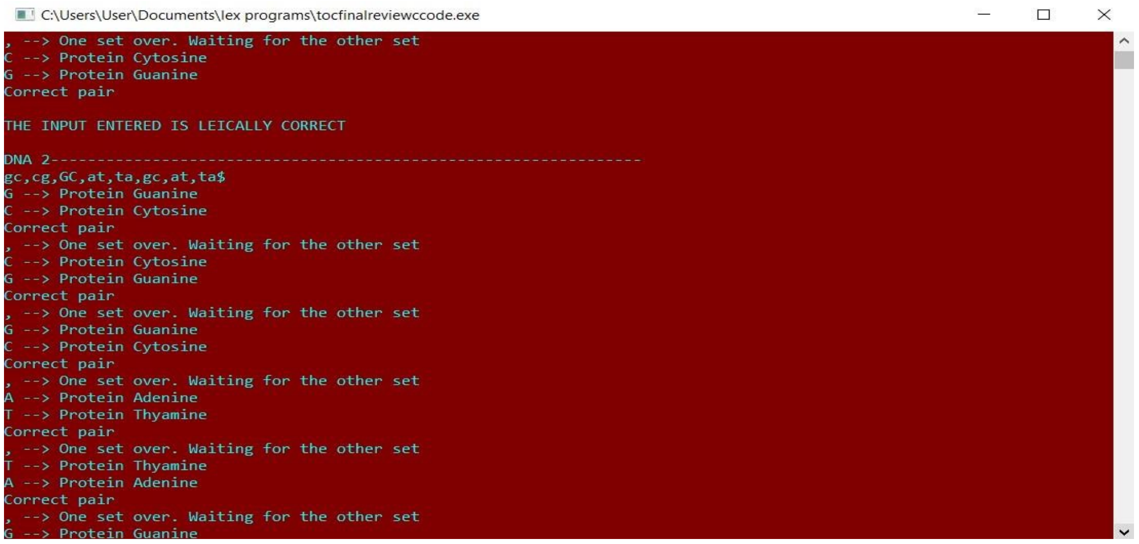


C:\Users\User\Documents\lex programs\tocfinalreviewccode.exe

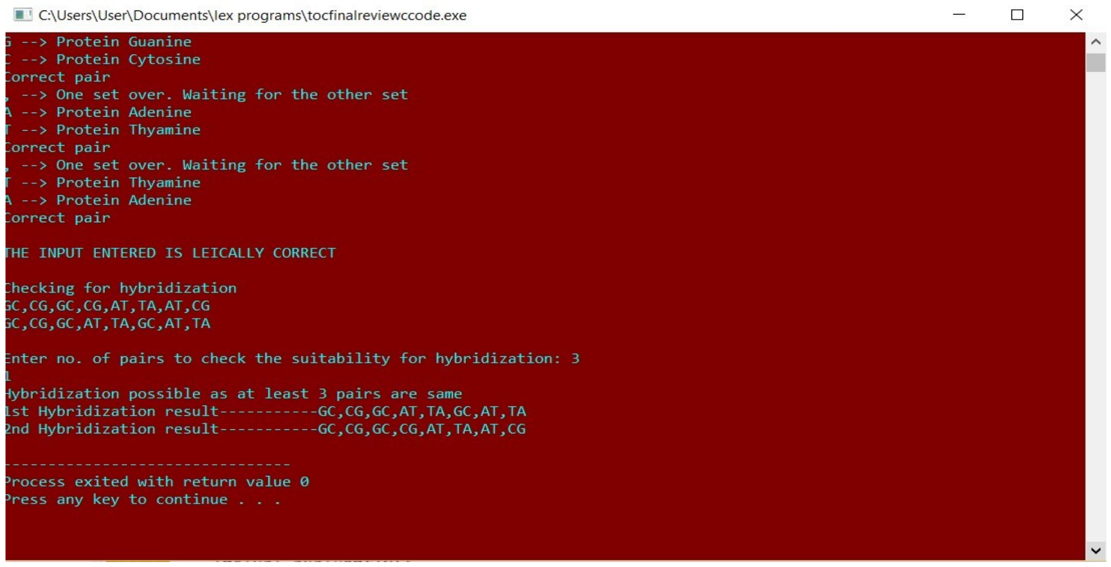
```
, --> One set over. Waiting for the other set
T --> Protein Thyamine
A --> Protein Adenine
Correct pair
, --> One set over. Waiting for the other set
G --> Protein Guanine
C --> Protein Cytosine
Correct pair
, --> One set over. Waiting for the other set
A --> Protein Adenine
T --> Protein Thyamine
Correct pair
, --> One set over. Waiting for the other set
T --> Protein Thyamine
A --> Protein Adenine
Correct pair

THE INPUT ENTERED IS LEICALLY CORRECT

Checking for hybridization
AT,TA,GC,CG,GC,TA,AT,CG
GC,CG,GC,AT,TA,GC,AT,TA

Enter no. of pairs to check the suitability for hybridization: 5
1
Hybridization not possible as at least 5 pairs are not same

--------------------------------
Process exited with return value 4294967295
Press any key to continue . . .
```


C:\Users\User\Documents\lex programs\tocfinalreviewccode.exe

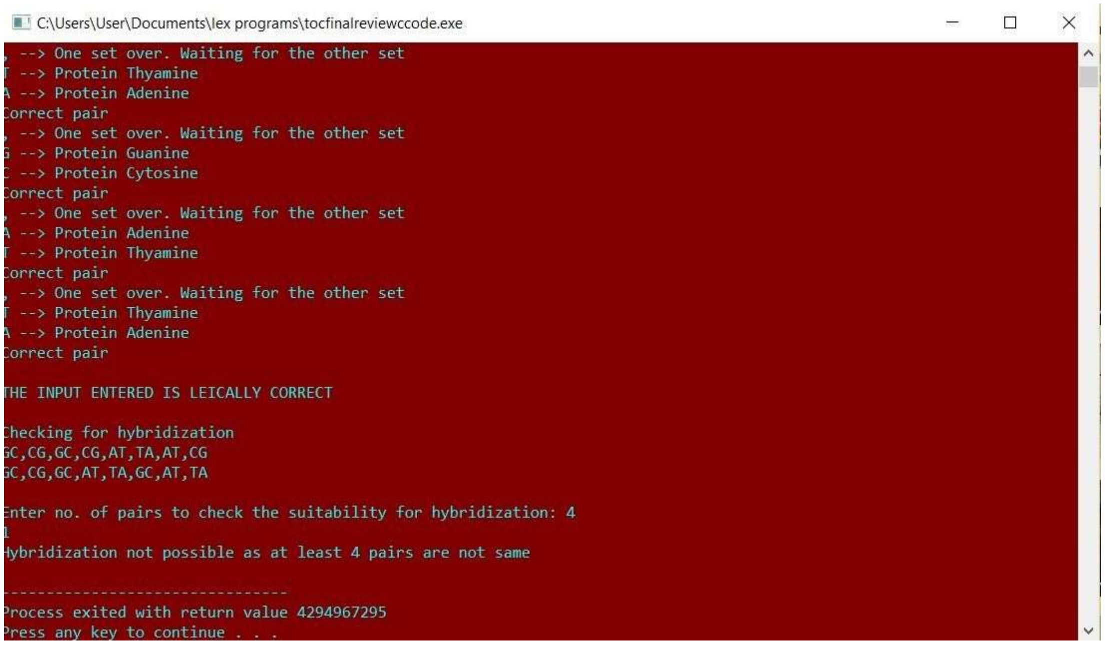
```
DNA 1----------------------------------------------------------------
gc,CG,gc,cg,at,ta,at,cg$
G --> Protein Guanine
C --> Protein Cytosine
Correct pair
, --> One set over. Waiting for the other set
C --> Protein Cytosine
G --> Protein Guanine
Correct pair
, --> One set over. Waiting for the other set
G --> Protein Guanine
C --> Protein Cytosine
Correct pair
, --> One set over. Waiting for the other set
C --> Protein Cytosine
G --> Protein Guanine
Correct pair
, --> One set over. Waiting for the other set
A --> Protein Adenine
T --> Protein Thyamine
Correct pair
, --> One set over. Waiting for the other set
T --> Protein Thyamine
A --> Protein Adenine
Correct pair
, --> One set over. Waiting for the other set
A --> Protein Adenine
T --> Protein Thyamine
Correct pair
, --> One set over. Waiting for the other set
```


C:\Users\User\Documents\lex programs\tocfinalreviewccode.exe

```
, --> One set over. Waiting for the other set
C --> Protein Cytosine
G --> Protein Guanine
Correct pair

THE INPUT ENTERED IS LEICALLY CORRECT

DNA 2----------------------------------------------------------------
gc,cg,GC,at,ta,gc,at,ta$
G --> Protein Guanine
C --> Protein Cytosine
Correct pair
, --> One set over. Waiting for the other set
C --> Protein Cytosine
G --> Protein Guanine
Correct pair
, --> One set over. Waiting for the other set
G --> Protein Guanine
C --> Protein Cytosine
Correct pair
, --> One set over. Waiting for the other set
A --> Protein Adenine
T --> Protein Thyamine
Correct pair
, --> One set over. Waiting for the other set
T --> Protein Thyamine
A --> Protein Adenine
Correct pair
, --> One set over. Waiting for the other set
G --> Protein Guanine
```


C:\Users\User\Documents\lex programs\tocfinalreviewccode.exe

```
G --> Protein Guanine
C --> Protein Cytosine
Correct pair
, --> One set over. Waiting for the other set
A --> Protein Adenine
T --> Protein Thyamine
Correct pair
, --> One set over. Waiting for the other set
T --> Protein Thyamine
A --> Protein Adenine
Correct pair

THE INPUT ENTERED IS LEICALLY CORRECT

Checking for hybridization
GC,CG,GC,CG,AT,TA,AT,CG
GC,CG,GC,AT,TA,GC,AT,TA

Enter no. of pairs to check the suitability for hybridization: 3
1
Hybridization possible as at least 3 pairs are same
1st Hybridization result-----------GC,CG,GC,AT,TA,GC,AT,TA
2nd Hybridization result-----------GC,CG,GC,CG,AT,TA,AT,CG

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```


C:\Users\User\Documents\lex programs\tocfinalreviewccode.exe

```
, --> One set over. Waiting for the other set
T --> Protein Thyamine
A --> Protein Adenine
Correct pair
, --> One set over. Waiting for the other set
G --> Protein Guanine
C --> Protein Cytosine
Correct pair
, --> One set over. Waiting for the other set
A --> Protein Adenine
T --> Protein Thyamine
Correct pair
, --> One set over. Waiting for the other set
T --> Protein Thyamine
A --> Protein Adenine
Correct pair

THE INPUT ENTERED IS LEICALLY CORRECT

Checking for hybridization
GC,CG,GC,CG,AT,TA,AT,CG
GC,CG,GC,AT,TA,GC,AT,TA

Enter no. of pairs to check the suitability for hybridization: 4
1
Hybridization not possible as at least 4 pairs are not same

--------------------------------
Process exited with return value 4294967295
Press any key to continue . . .
```

6. CONCLUSION

The conclusion we can draw is that our program is capable of checking any DNA sequence and to check whether hybridization is possible or not. Also in the end we can conclude this application can be very useful for analyzing DNA and hybridization purpose in the Biology and forensics lab and this program can be further developed for more interactive application.

7. REFERENCES

The Go Programming Language Paperback by Alan A. A. Donovan , Brian W. Kernighan

Molecular Biology Paperback by Rastogi S.C.