MSc. (Computer Science) Semester - I

Name: Shubham Lad **Roll No: 512**

Paper III (Bioinformatics)

INDEX

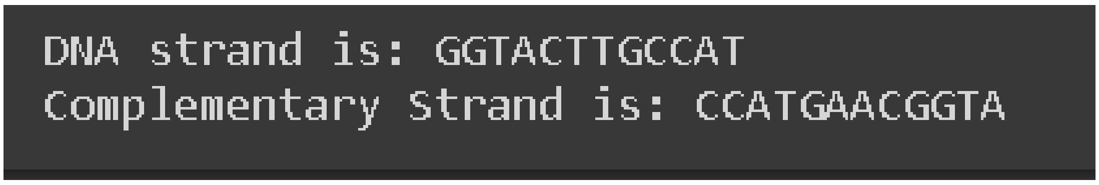
Practical 1

Aim: Complementary DNA Sequence.

Code:

```
def complementary_strand_find(dna_strand):
    complementary_strand = ""
    for base in dna_strand:
        if base == "A":
            complementary_strand += "T"
        elif base == "T":
            complementary_strand += "A"
        elif base == "U":
            complementary_strand += "A"
        elif base == "G":
            complementary_strand += "C"
        elif base == "C":
            complementary_strand += "G"
        elif base == "Y":
            complementary_strand += "R"
        elif base == "R":
            complementary_strand += "Y"
        else:
            print("Wrong input")
            complementary_strand = None
            break
    return complementary_strand
if __name__ == "__main__":
  dna_strand = "GGTACTTGCCAT"
print("DNA strand is:",dna_strand)
print("Complementary Strand is:",complementary_strand_find(dna_strand))
```

Output:



```
DNA strand is: GGTACTTGCCAT
Complementary Strand is: CCATGAACGGTA
```

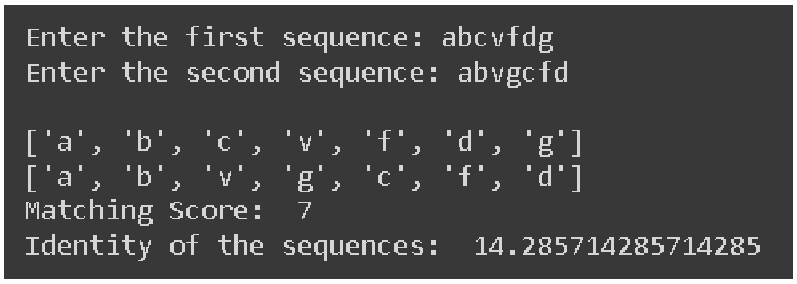
Practical 2

Aim: Identity of Two protein sequence

Code:

```
se1=input("Enter the first sequence: ")
se2=input("Enter the second sequence: ")
seq1=list(se1)
seq2=list(se2)
def find_identity(a,b):
  gap(a,b)
  print(a)
  print(b)
  score=0
  length=len(a)
  total_elements=len(a)*len(b)
  for i in range(0,length):
    for j in range(0,length):
      if(a[i]==b[j]):
        score=score+1
  identity=(score/total_elements)*100
  print("Matching Score: ",score)
  print("Identity of the sequences: ",identity)
def gap(a,b):
  if(len(a)==len(b)):
    print()
  else:
    k=int(input("enter the position to insert gap: "))
    if (len(a)<len(b)):
      a.insert(k,'-')
    else:
      b.insert(k,'-')
  return(a,b)
find_identity(seq1,seq2)
```

Output:



```
Enter the first sequence: abcvfdg
Enter the second sequence: abvgcfd

['a', 'b', 'c', 'v', 'f', 'd', 'g']
['a', 'b', 'v', 'g', 'c', 'f', 'd']
Matching Score:  7
Identity of the sequences:  14.285714285714285
```

Practical 3

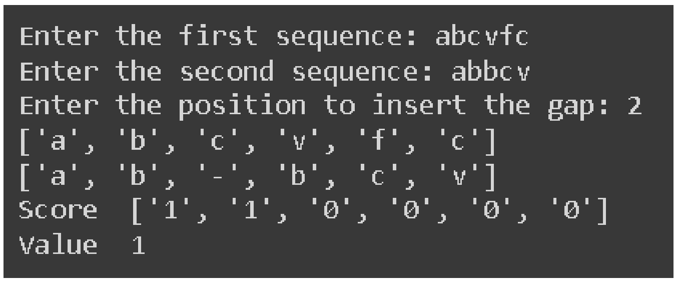
Aim: Pairwise Sequence Alignment

Code:

```
seq1=list(input("Enter the first sequence: "))
seq2=list(input("Enter the second sequence: "))
score=[]
def pairwise(a,b):
    gap(a,b)
    print(a)
    print(b)
    value=1
    length=len(a)
    for i in range(0,length):
        if a[i]==b[i]:
            score.append('1')
            value+=0
        else:
            score.append('0')
    print("Score ",score)
    print("Value ",value)

def gap(a,b):
    if (len(a)==len(b)):
        print()
    else:
        k=int(input("Enter the position to insert the gap: "))
        if (len(a)<len(b)):
            a.insert(k,'-')
        else:
            b.insert(k,'-')
        return (a,b)
pairwise(seq1,seq2)
```

Output:



```
Enter the first sequence: abcvfc
Enter the second sequence: abbcv
Enter the position to insert the gap: 2
['a', 'b', 'c', 'v', 'f', 'c']
['a', 'b', '-', 'b', 'c', 'v']
Score  ['1', '1', '0', '0', '0', '0']
Value  1
```

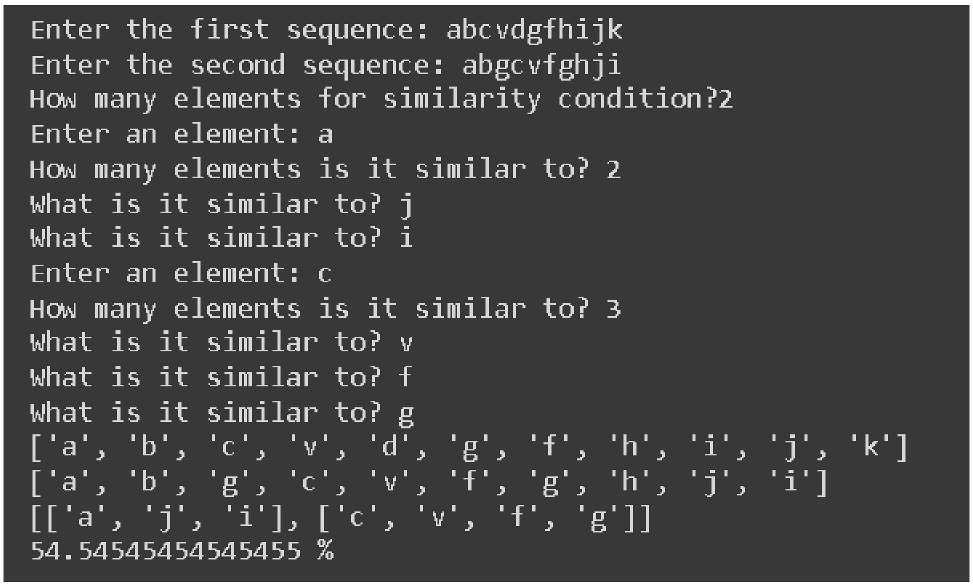
Practical 4

Aim: Similarity between two protein sequence

Code:

```
sequence_one=input("Enter the first sequence: ")
sequence_two=input("Enter the second sequence: ")
how_many=int(input("How many elements for similarity condition?"))
similarities=[]
for i in range(0,how_many):
  a=input("Enter an element: ")
  c=int(input("How many elements is it similar to? "))
  similarities.append([])
  similarities[i].append(a)
  for j in range(0,c):
    b=input("What is it similar to? ")
    similarities[i].append(b)
def compare(o,t,s):
  print(o)
  print(t)
  print(s)
#checking if similar
  score=0
  for i in range(len(o)):
      for j in range(len(s)):
        if o[i] in s[j] and t[i] in s[j] and o[i] != t[i]:
          score+=1
        #calculating similarity
  similarity= (score*100)/len(o)
  print("The similarity is: ",similarity)
print(compare(list(sequence_one),list(sequence_two),similarities),"%")
```

Output:



```
Enter the first sequence: abcvdgfhijk
Enter the second sequence: abgcvfghji
How many elements for similarity condition?2
Enter an element: a
How many elements is it similar to? 2
What is it similar to? j
What is it similar to? i
Enter an element: c
How many elements is it similar to? 3
What is it similar to? v
What is it similar to? f
What is it similar to? g
['a', 'b', 'c', 'v', 'd', 'g', 'f', 'h', 'i', 'j', 'k']
['a', 'b', 'g', 'c', 'v', 'f', 'g', 'h', 'j', 'i']
[['a', 'j', 'i'], ['c', 'v', 'f', 'g']]
54.54545454545455 %
```

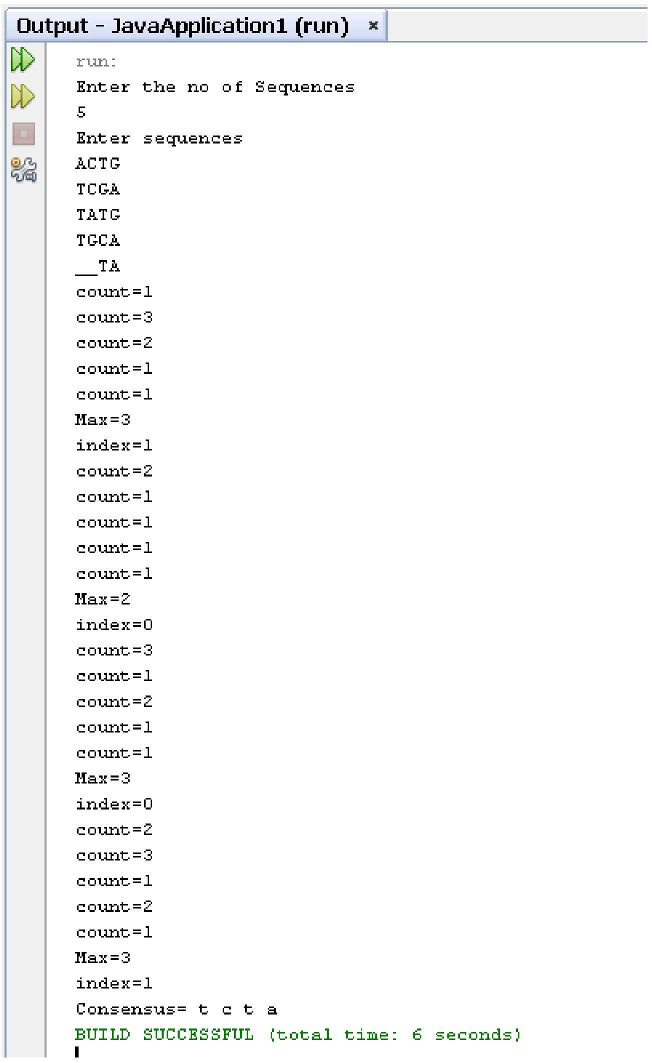
Practical 5

Aim: Multiple Sequence Alignment

Code: prac_5.java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;
import java.util.Set;
public class prac_5{
public static void main(String str[]) throws IOException
 {
 int n, i,j,k,count;
 String seq[],cons[];
 ArrayList<Integer> a = new ArrayList<Integer>();
 ArrayList s = new ArrayList();
 BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
 System.out.println("Enter the no of Sequences");
 n=Integer.parseInt(br.readLine());
 seq=new String[n];
 System.out.println("Enter sequences");
 for(i=0;i<n;i++)
 seq[i]=br.readLine();
 cons=new String[seq[0].length()];
 for(j=0;j<seq[0].length();j++)
 cons[j]=" ";
 for(j=0;j<seq[0].length();j++)
 {
 a.clear();
 s.clear();
 for(i=0;i<n;i++)
 {
 count=1;
 for(k=i+1;k<n;k++)
 {
 if(seq[i].charAt(j)==seq[k].charAt(j))
 count++;
 }
 System.out.println("count="+count);
 a.add(count);
 s.add(seq[i].charAt(j));
 }
 Set<String> set = new HashSet<>(s);
 ArrayList setlist = new ArrayList(set);
Collections.sort(setlist);
 if (setlist.contains('-') &&setlist.size()==2){
 cons[j]+="-"+setlist.get(1);
 }
 else if (setlist.size()==1){
 cons[j]+="-"+setlist.get(0);
 }
 else{
 int m = Collections.max(a);
 int index=a.indexOf(m);
 System.out.println("Max="+m);
 cons[j]+=s.get(index);
 System.out.println("index="+index);
 for(i=index+1;i<a.size();i++)
 {
 if(a.get(i)==m)
 cons[j]+="/"+s.get(i);
 }
 }
 }
 System.out.println("Consensus=");
 for(j=0;j<seq[0].length();j++){
 /**Updated Snippet 2**/
 if(cons[j].length()==2)
 System.out.print(cons[j].toLowerCase());
 else if(cons[j].length()==3)
 System.out.print(cons[j].replace("-",""));
 else
 System.out.print(cons[j]);
 }
 }
}
```

Output:


Output - JavaApplication1 (run)

```
run:
Enter the no of Sequences
5
Enter sequences
ACTG
TCGA
TATG
TGCA
__TA
count=1
count=3
count=2
count=1
count=1
Max=3
index=1
count=2
count=1
count=1
count=1
count=1
Max=2
index=0
count=3
count=1
count=2
count=1
count=1
Max=3
index=0
count=2
count=3
count=1
count=2
count=1
Max=3
index=1
Consensus= t c t a
BUILD SUCCESSFUL (total time: 6 seconds)
```

Practical 6

Aim: Motif Finding

Code:

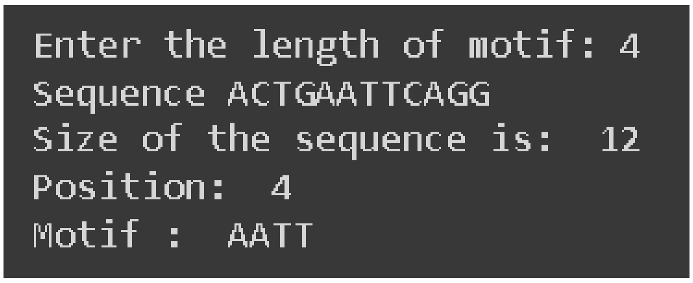
prac_6.py

```
import random
l=int(input("Enter the length of motif: "))
file=open("mot.txt","r")
r=file.read()
print("Sequence",r)
size=len(r)
print("Size of the sequence is: ",size)
pos=random.randint(0,len(r)-5)
#pos=1
print("Position: ",pos)
motif=r[pos:pos+l]
print("Motif : ",motif)
i=pos+1
while(i<=size-1):
    if(motif==r[i:i+1]):
        str1=r[i:i+1]
        print("Match motif",str1)
        file1=open("motoutput.txt","a")
        file1.write(str1+" ")
    i+=1
```

mot.txt

```
ACTGAATTCAGG
```

Output:



```
Enter the length of motif: 4
Sequence ACTGAATTCAGG
Size of the sequence is:  12
Position:  4
Motif :  AATT
```

Practical 7

Aim: Perform a BLAST search on any gene sequence and write a code to count the no of repetition of each nucleotide in the sequence.

Code:

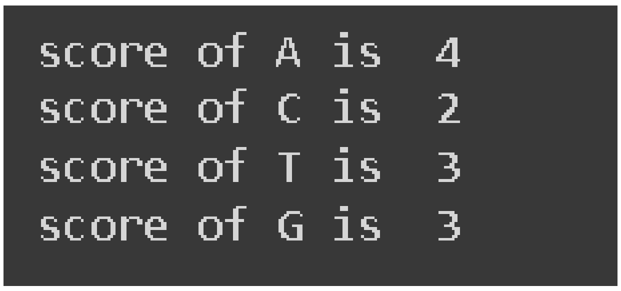
prac_7.py

```
file=open("gene.txt","r")
r=file.read()
size=len(r)
score_A=0
score_C=0
score_T=0
score_G=0
for i in range(size):
  if(r[i]=='A'):
    score_A+=1
  elif (r[i]=='C'):
    score_C+=1
  elif (r[i]=='T'):
    score_T+=1
  elif (r[i]=='G'):
    score_G+=1
print("score of A is ",score_A)
print("score of C is ",score_C)
print("score of T is ",score_T)
print("score of G is ",score_G)
```

gene.txt

```
ACTGAATTCAGG
```

Output:



```
score of A is  4
score of C is  2
score of T is  3
score of G is  3
```

Practical 8

Aim: Regular Expression

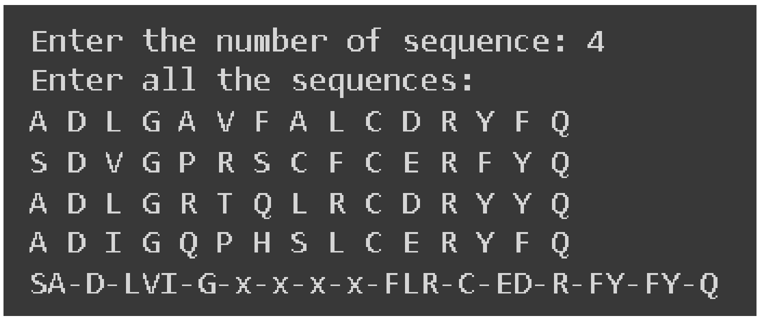
Code:

```python
def gen_reg_exp(seq_list, no_of_col):
    final_list=[]
    for colnum in range(no_of_col):
        collist=[]
        for colseq in seq_list:
            collist.append(colseq[colnum])
        if len(set(collist))==len(collist):
            #print(final_list)
            final_list.append('x')
        else:
            if len(set(collist))==1:
                final_list.append(collist[0])
            else:
                final_list.append(''.join(set(collist)))
    display_output(final_list)

def display_output(final_list):
    print(*final_list, sep='-')

no_of_seq=int(input("Enter the number of sequence: "))
print("Enter all the sequences: ")
seq_list=[]
for _ in range(no_of_seq):
    seq_list.append(list(map(str, input("").split())))
gen_reg_exp(seq_list, len(seq_list[0]))
```

Output:



```
Enter the number of sequence: 4
Enter all the sequences:
A D L G A V F A L C D R Y F Q
S D V G P R S C F C E R F Y Q
A D L G R T Q L R C D R Y Y Q
A D I G Q P H S L C E R Y F Q
SA-D-LVI-G-x-x-x-x-FLR-C-ED-R-FY-FY-Q
```

Practical 9

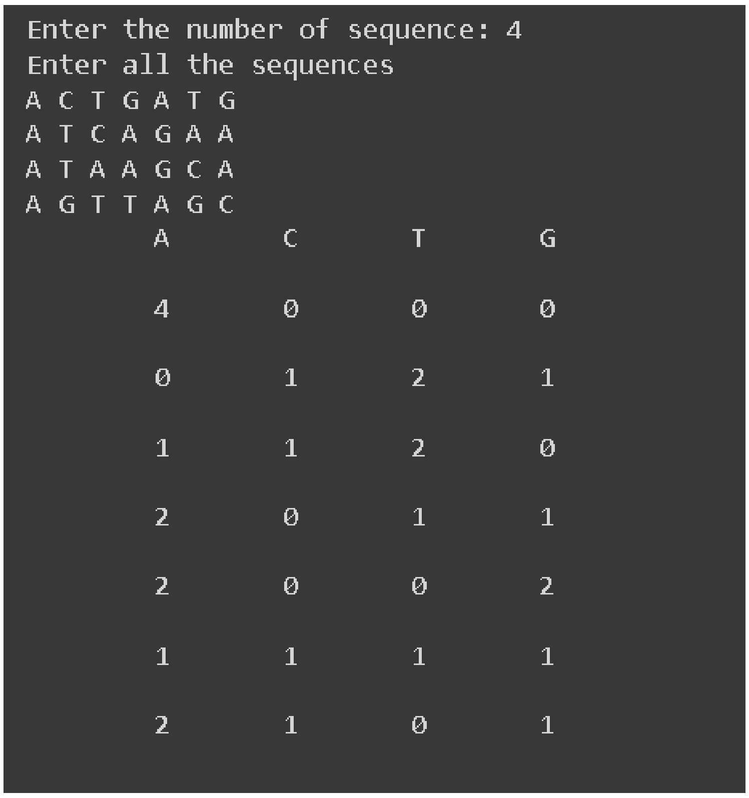
Aim: Enter six protein sequence of different organism and write a program to find a fingerprint of sequence.

Code:

```
def solve_fingerprint(seq_list, no_of_col):
  seq_dict=dict()
  for colnum in range(no_of_col):
      counta,countc,countt,countg=0,0,0,0
      for colseq in seq_list:
        if colseq[colnum]=='A':
          counta+=1
        elif colseq[colnum]=='T':
          countt+=1
        elif colseq[colnum]=='C':
          countc+=1
        elif colseq[colnum]=='G':
          countg+=1
      seq_dict[colnum]=[counta,countc,countt,countg]
  display_results(seq_dict)

def display_results(seq_dict):
  print("\tA \tC \tT \tG")
  for key in seq_dict:
    print("\n",*seq_dict[key],sep="\t")
no_of_seq=int(input("Enter the number of sequence: "))
print("Enter all the sequences")
seq_list=[]
for _ in range(no_of_seq):
  seq_list.append(list(map(str, input("").split())))
solve_fingerprint(seq_list,len(seq_list[0]))
```

Output:



```
Enter the number of sequence: 4
Enter all the sequences
A C T G A T G
A T C A G A A
A T A A G C A
A G T T A G C
        A       C       T       G

        4       0       0       0

        0       1       2       1

        1       1       2       0

        2       0       1       1

        2       0       0       2

        1       1       1       1

        2       1       0       1
```