

**Practical No: 1**

**Aim:** Write a Python/Java code to perform pairwise alignment. Take 2 sequences from user and calculate the score.

**Code:**

```
se1=input("Enter the first sequence::")
se2=input("Enter the second sequence::")
seq1=list(se1)
seq2=list(se2)
score=[]
```

```
def Pairwise_alignment(a,b):
```

```
    gap(a,b)
    print(a)
    print(b)
    value=0
    length=len(a)
    for i in range(0,length):
        if(a[i]==b[i]):
            score.append('1')
            value=value+1
        else:
            score.append('0')
    print(score)
    print(value)
```

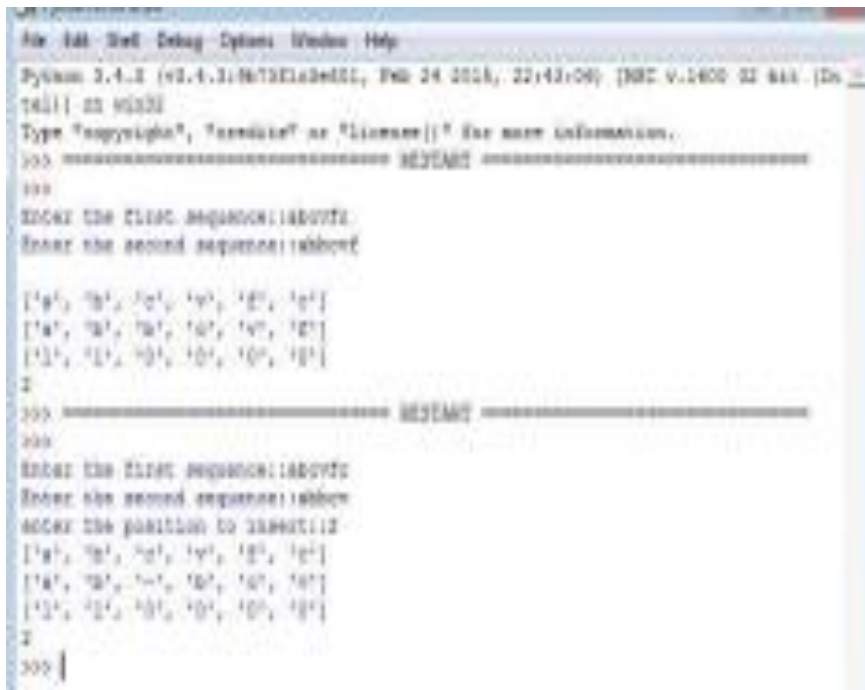
```
def gap(a,b):
```

```
    if(len(a)==len(b)):
        print()
    else:
```

```
k=int(input("enter the position to insert:"))  
if (len(a)<len(b)):  
    a.insert(k,'-')  
else:  
    b.insert(k,'-')  
return(a,b)
```

Pairwise\_alignment(seq1,seq2)

Output:



```
File Edit Shell Debug Options Window Help  
Python 3.4.2 (43.4.316732163216321, Feb 24 2016, 22:43:04) [AMD64] on win32  
Type "copyright", "credits()" or "license()" for more information.  
>>> ===== RESTART =====  
>>>  
Enter the first sequence:abcde  
Enter the second sequence:abbcde  
  
['a', 'b', '-', 'c', 'd', 'e']  
['a', 'b', '-', 'b', 'c', 'd', 'e']  
['a', 'b', '-', 'b', 'c', 'd', 'e']  
I  
>>> ===== RESTART =====  
>>>  
Enter the first sequence:abcde  
Enter the second sequence:abbcde  
Enter the position to insert:2  
['a', 'b', '-', 'c', 'd', 'e']  
['a', 'b', '-', 'b', 'c', 'd', 'e']  
['a', 'b', '-', 'b', 'c', 'd', 'e']  
I  
>>> |
```

## Practical No: 2

Aim: Write a Python/Java code to find the identity value of a given sequences. Take the sequence from user.

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:

```
Python 3.4.3 Shell
File Edit Shell Debug Options Window Help
Python 3.4.3 (v3.4.3:0b75f13e601, Feb 24 2015, 22:43:04) [MSC v.1600 64-bit]
Type "copyright", "credits()" or "license()" for more >>>
>>>
Enter the first sequence:abcdfg
Enter the second sequence:abcefgd
['a', 'b', 'c', 'd', 'e', 'f', 'g']
['a', 'b', 'c', 'e', 'f', 'd', 'g']
Matching Score: 7
Identity of the sequences: 16.285714285714285
>>>
Enter the first sequence:abcdfg
Enter the second sequence:abcTga
enter the position to insert:8
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'a']
['a', 'b', 'c', 'e', 'f', 'd', 'g', 'a']
Matching Score: 8
Identity of the sequences: 12.8588956213685
>>>
```

## Practical No: 3

Aim: Write a Python/Java code to find the Similarity value of a given sequences. Take the sequence from user.

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
```

```
print(compare(list(sequence_one),list(sequence_two),similarities,"%")
```

[illegible]

## Practical No: 4

Aim: Enter genome of five different organism and write a python/java program to find consensus sequence using Multiple Sequence Alignment (MSA) technique.

Code:

```
import java.io.*;
import java.util.*;
public class Consensus
{
    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));

}

/**Updated Snippet 1**/

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);

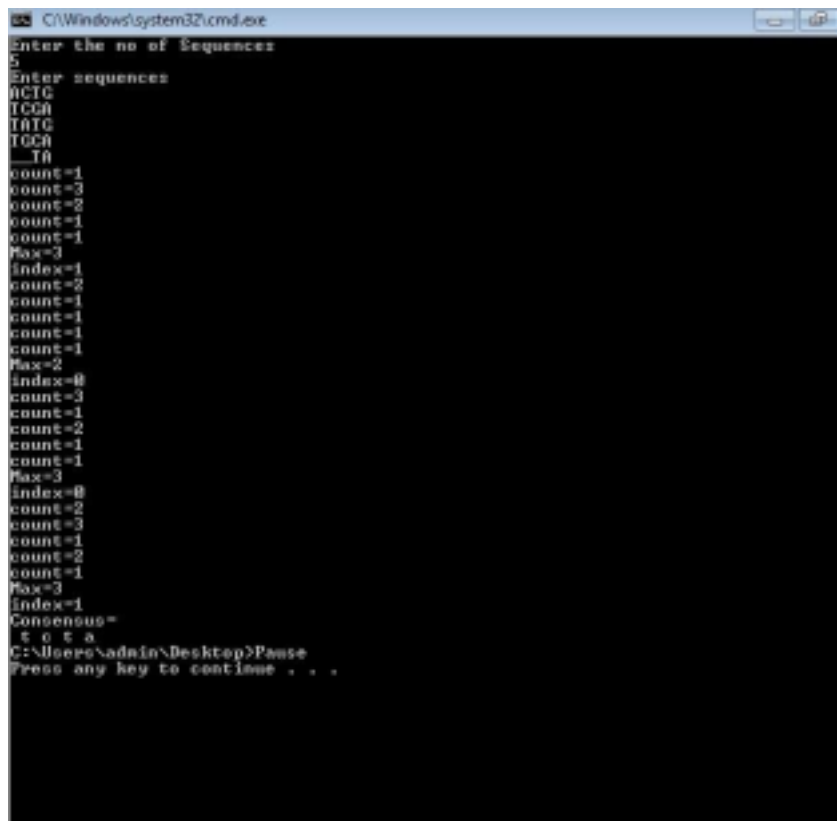
    System.out.println("index="+index);

    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:



```
C:\Windows\system32\cmd.exe
Enter the no of Sequences
5
Enter sequences
ACTG
TCGA
TATG
TCGA
TGA
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 G T A
C:\Users\admin\Desktop>Pause
Press any key to continue . . .
```

## Practical No: 5

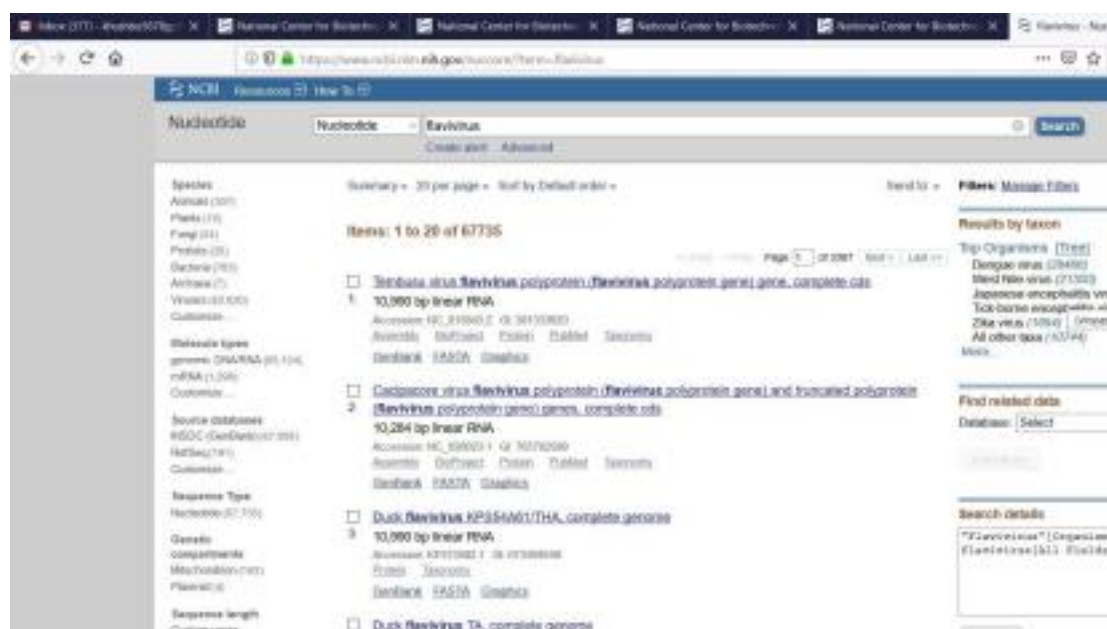
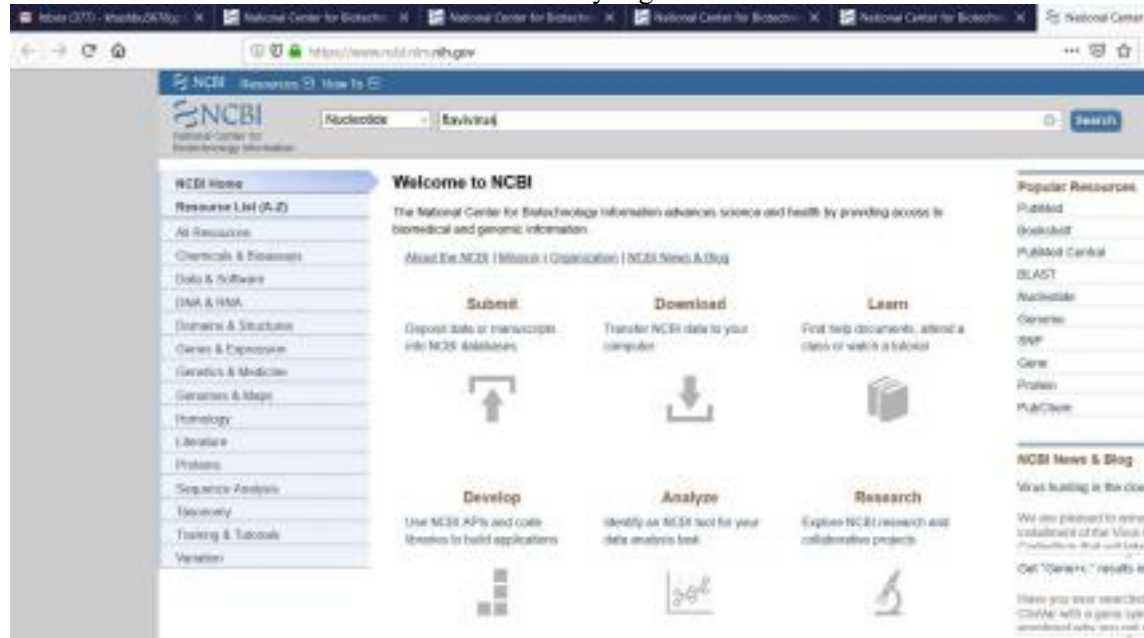
**Aim:** Perform a BLAST search on a specific gene sequence of a specify organism.

**Steps:**

Go to the National Center for Biotechnology Information Site

<https://www.ncbi.nlm.nih.gov/>

Select Nucleotide from All Databases and find any organism in a search bar.



The screenshot displays the NCBI GenBank entry for the Tembusu virus flavivirus polyprotein (flavivirus polyprotein gene) gene, complete cds. The entry is identified by the accession number NC\_015643.2. The sequence is 10000 bp long, with a linear structure. The entry includes a detailed description of the gene structure and sequence, as well as a list of references and authors.

**Accession:** NC\_015643.2  
**Version:** NC\_015643.2  
**Definition:** Tembusu virus flavivirus polyprotein (flavivirus polyprotein gene) gene, complete cds.  
**Accession:** NC\_015643.2  
**Version:** NC\_015643.2  
**Definition:** Tembusu virus flavivirus polyprotein (flavivirus polyprotein gene) gene, complete cds.  
**Accession:** NC\_015643.2  
**Version:** NC\_015643.2  
**Definition:** Tembusu virus flavivirus polyprotein (flavivirus polyprotein gene) gene, complete cds.

Run BLAST option we have to select

The screenshot shows the NCBI BLAST web interface. The browser address bar displays the URL: [https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE=Proteinblast&PROGRAM=blastn&NC\\_015943.2%3AAB432+new](https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE=Proteinblast&PROGRAM=blastn&NC_015943.2%3AAB432+new).

**Align two or more sequences (p)**

**Choose Search Set**

**Database:** ☐ Human genomic + transcript, ☐ Mouse genomic + transcript, ☒ Others (or etc.)

**Organisms:** ☐ Nucleotide collection (nr/nt)

**Exclude:** ☐ Masked regions (or all noncoding regions will be excluded), ☐ Exclude

**Limit to:** ☐ Sequences from type material

**Enter query:**

**Program Selection:**

**Optimize for:**

☒ Highly similar sequences (optimized)

☐ More dissimilar sequences (discontinuous optimized)

☐ Somewhat similar sequences (blastn)

☐ Choose a BLAST algorithm (p)

**BLAST** Search database Nucleotide collection (nr/nt) using Megablast (Optimize for highly similar sequences)

☐ Show results in a new window

[\(p\) Adjust parameters](#)

BLAST is a registered trademark of the National Library of Medicine

The screenshot shows the NCBI BLAST results page. The browser address bar displays the URL: <https://blast.ncbi.nlm.nih.gov/Blast.cgi>.

**Sequences producing significant alignments**

☒ select all 39 sequences selected

Description	Max Score	Total Score	Query Cover	E value
<input checked="" type="checkbox"/> Terhuya virus strain JS201, complete genome	29086	29086	99%	0.0
<input checked="" type="checkbox"/> Terhuya virus strain JS2010, complete genome	29086	29086	99%	0.0
<input checked="" type="checkbox"/> Duck egg-drop syndrome virus strain DED, complete genome	29045	29045	99%	0.0
<input checked="" type="checkbox"/> Terhuya virus isolate Terhuya virus strain, complete genome	29000	29000	99%	0.0
<input checked="" type="checkbox"/> Duck Terhuya virus isolate #2, complete genome	29000	29000	99%	0.0
<input checked="" type="checkbox"/> Duck egg-drop syndrome virus strain JXSP, complete genome	29000	29000	99%	0.0
<input checked="" type="checkbox"/> Terhuya virus isolate YY5, complete genome	29015	29015	99%	0.0
<input checked="" type="checkbox"/> Terhuya virus isolate JDM5, complete genome	29009	29009	99%	0.0
<input checked="" type="checkbox"/> Terhuya virus isolate J2-6, complete genome	29009	29009	99%	0.0
<input checked="" type="checkbox"/> Terhuya virus strain AH-F-50 from China, complete genome	29004	29004	99%	0.0
<input checked="" type="checkbox"/> Duck egg-drop syndrome virus strain pgsn1, complete genome	29004	29004	99%	0.0
<input checked="" type="checkbox"/> Terhuya virus genome TR45, complete genome, strain TSH/NC/YY10p	19098	19098	99%	0.0
<input checked="" type="checkbox"/> Duck Terhuya virus strain SZ_2016, complete genome	19098	19098	99%	0.0
<input checked="" type="checkbox"/> Duck egg-drop syndrome virus strain BAW, complete genome	19098	19098	99%	0.0
<input checked="" type="checkbox"/> Duck Terhuya virus strain GGU01, complete genome	19098	19098	99%	0.0

Here the result will be display

Downloaded GenBank Graphics

**Tembusu virus strain JS804, complete genome**  
 Sequence ID: [JF895933.3](#) Length: **10990** Number of Matches: **1**

Range: 1: 1 to 10990 [Sequence](#) [Graphics](#)

Score: 28295 bits(10990) Expect: 0.0 Identities: 10990/10990(100%) Gaps: 0/10990(0%) Mismatches: 0/10990(0%)

Query	Subject	Score	Expect	Identities	Gaps	Mismatches
Query 1	AGAGTTGCGCTGTGTGAACCTATTCGAAACGCTTTTGGAGTATGCGCTGTGAACCTAA	60				
Subject 1	AGAGTTGCGCTGTGTGAACCTATTCGAAACGCTTTTGGAGTATGCGCTGTGAACCTAA	60				
Query 61	ACACATTTTGAAGCTTTTGTATATGACCACTATTTCTAAAGAAAGCCGAAAGGAC	120				
Subject 61	ACACATTTTGAAGCTTTTGTATATGACCACTATTTCTAAAGAAAGCCGAAAGGAC	120				
Query 121	CGGCTGAGGCGGCTTTTCTCAATATCTAAAGCGGAGCGCTCGGCGAAATCGGCTAGC	180				
Subject 121	CGGCTGAGGCGGCTTTTCTCAATATCTAAAGCGGAGCGCTCGGCGAAATCGGCTAGC	180				
Query 181	CGGATAAAGGAGCGATCGATCGGCTCGGAGGAGCGAGCGATAGGCTTTGCTGCT	240				
Subject 181	CGGATAAAGGAGCGATCGATCGGCTCGGAGGAGCGAGCGATAGGCTTTGCTGCT	240				
Query 241	CGGCTGAGGCGGCTTTTCTCAATATCTAAAGCGGAGCGCTCGGCGAAATCGGCTAGC	300				
Subject 241	CGGCTGAGGCGGCTTTTCTCAATATCTAAAGCGGAGCGCTCGGCGAAATCGGCTAGC	300				
Query 361	ATGAGGCTGCTTTTGAATTAATGAGGAGCGAGGCTGCTGAGGAGCTTCAAGCTTGCAT	360				
Subject 361	ATGAGGCTGCTTTTGAATTAATGAGGAGCGAGGCTGCTGAGGAGCTTCAAGCTTGCAT	360				
Query 420	TGGACATGCTCTGACCGACTGATTAAGCGGAGGCGGAGGCTGCTGAGGAGCTTCAAGCTTGCAT	420				
Subject 420	TGGACATGCTCTGACCGACTGATTAAGCGGAGGCGGAGGCTGCTGAGGAGCTTCAAGCTTGCAT	420				

**Related Info**  
[Gene association](#)

## Practical No: 6

Aim: Write a Python/Java code to find motif in a given sequence.

Code:

```
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",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
```

Output:

Enter the length of motif4

Sequence AGAAGTTCGAGAAGCCGTAGT

Size of the sequence 21

Position 0

Motif AGAA



## Practical No: 7

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

Code:

```
file=open("genes.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)
```

Output:

score of A is 6

score of C is 4

score of T is 7

score of G is 6

## Practical No: 8

Aim: Generate a regular expression enter three protein sequence of three different organism. Write Python/Java code to find regular expression for this sequences.

Code:

```
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

ADLGAVFALCDRYFQ

SDVGPRSCFCERFYQ

ADLGRTQLRCDRYYQ

ADIGQPHSLCERYFQ

SA-D-IVL-G-x-x-x-x-FRL-C-ED-R-YF-YF-Q

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:

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

2 1 0 1