T.C.
MARMARA UNIVERSITY

FACULTY of ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

CSE4074 Computer Networks

Project#2

Büşra YAŞAR-150114063

Emine Feyza Memiş-150114077

Hale ŞAHİN-150116841

INTRODUCTION

We worked on a particular region in a genome, so we only took some part of the genome as input. This input defined as one line of text in a file where the text contained only A, T, G or C. In project document, input file has 10 lines, where each line contains strings of length 500.

Our objective is to search for motifs and try to find the consensus string by using Randomized Motif Search and Gibbs Sampler.

After we found the consensus strings, we compared the scores and consensus strings obtained for different k values and gave it as output.

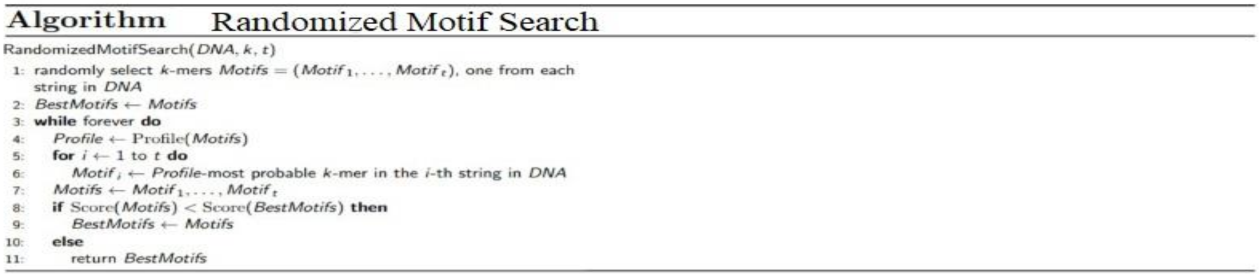
RANDOMIZED ALGORITHMS

Randomized algorithms make random rather than deterministic decisions. The main advantage is that no input can reliably produce worst-case results because the algorithm runs differently each time. These algorithms are commonly used in situations where no exact and fast algorithm is known. There are 5 different most used randomized algorithms:

- Randomized Quick-Sort
- Randomized Algorithms
- Greedy Profile Motif Search
- Gibbs Sampler
- Random Projections

RANDOMIZED MOTIF SEARCH

Randomized Motif Search may change all t motifs in Motifs = (Motif_1,…,Motif_t) in a single iteration. This strategy may prove reckless, since some correct motifs may potentially be discarded at the next iteration.

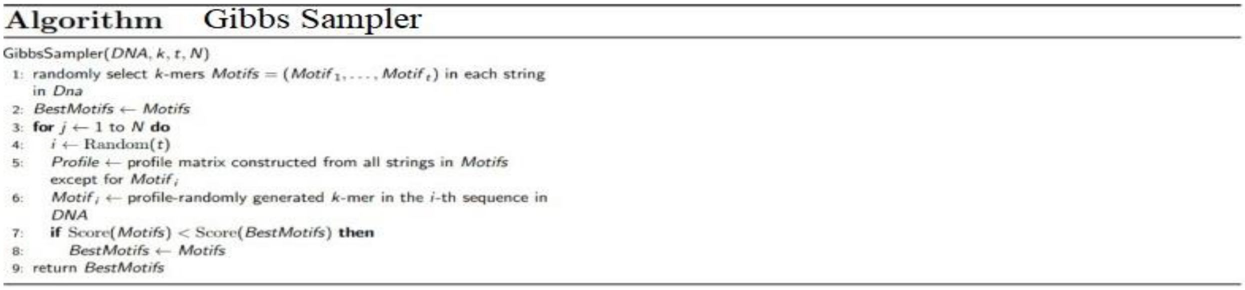


```
Algorithm    Randomized Motif Search
RandomizedMotifSearch($DNA, k, t$)
randomly select $k$-mers $Motifs = (Motif_1, \ldots, Motif_t)$, one from each
string in $DNA$
$BestMotifs \leftarrow Motifs$
while forever do
    $Profile \leftarrow \text{Profile}(Motifs)$
    for $i \leftarrow 1$ to $t$ do
        $Motif_i \leftarrow Profile$-most probable $k$-mer in the $i$-th string in $DNA$
    $Motifs \leftarrow Motif_1, \ldots, Motif_t$
    if $\text{Score}(Motifs) < \text{Score}(BestMotifs)$ then
        $BestMotifs \leftarrow Motifs$
    else
        return $BestMotifs$
```

(Figure-1: Pseudocode of Randomized Motif Search)

GIBBS SAMPLER

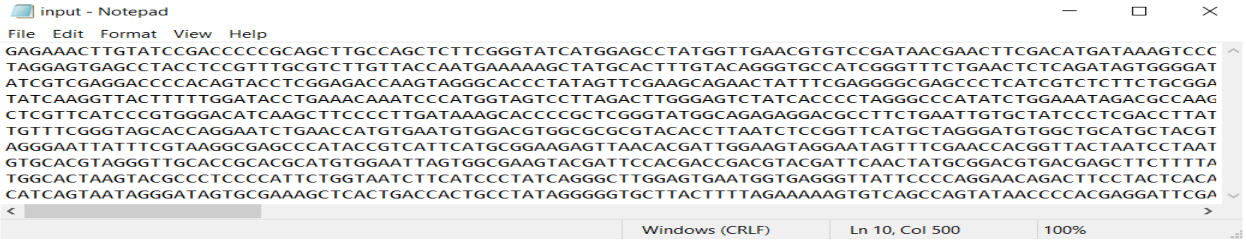
Gibbs Sampler is a more cautious iterative algorithm that discards a single k-mer from the current set of motifs at each iteration and decides to either keep it or replace it with a new one. It starts with randomly chosen k-mers in each of t DNA sequences, but it makes a random rather than a deterministic choice at each iteration. It uses randomly selected k-mers (Motif_1, …, Motif_t) to come up with another (hopefully higher scoring) set of k-mers.It randomly selects an integer i between 1 and t and then randomly changes only one Motif_i.



```
Algorithm  Gibbs Sampler
GibbsSampler(DNA, k, t, N)
  randomly select k-mers Motifs = ($Motif_1, \ldots, Motif_t$) in each string
  in Dna
  BestMotifs ← Motifs
  for j ← 1 to N do
      i ← Random(t)
      Profile ← profile matrix constructed from all strings in Motifs
      except for $Motif_i$
      $Motif_i$ ← profile-randomly generated k-mer in the i-th sequence in
      DNA
      if Score(Motifs) < Score(BestMotifs) then
          BestMotifs ← Motifs
  return BestMotifs
```

(Figure-2: Pseudocode of Gibbs Sampler)

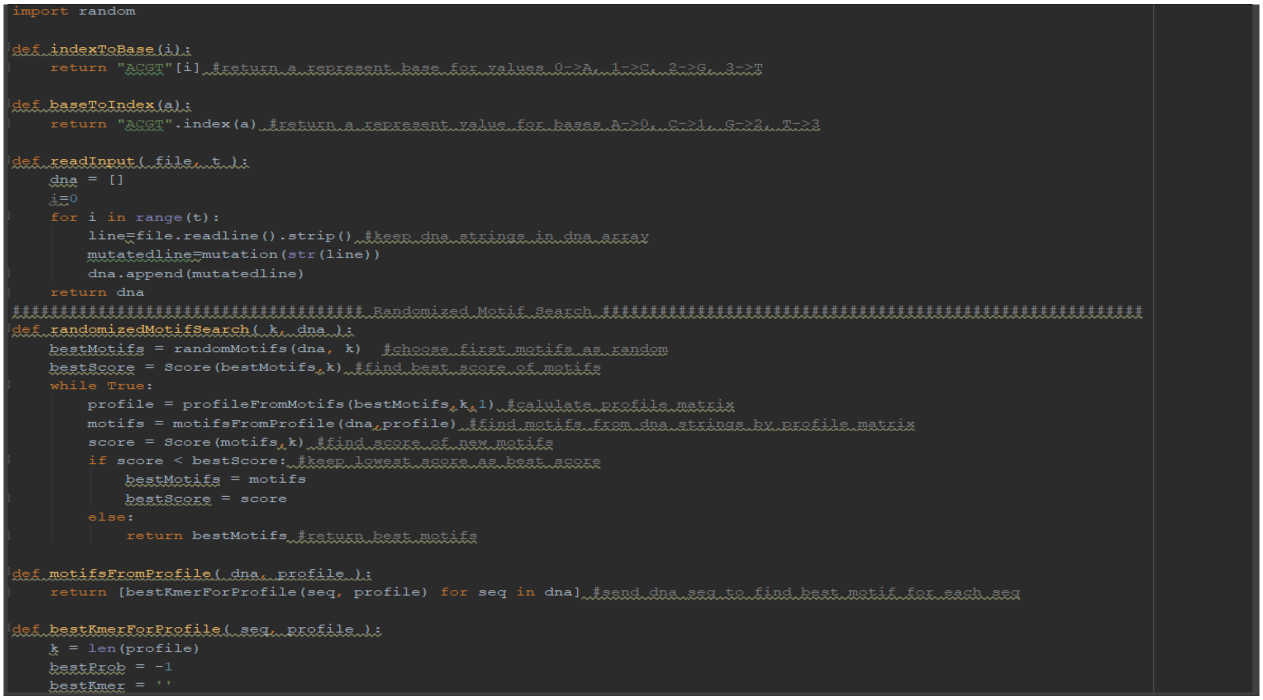
CODE AND OUTPUTS



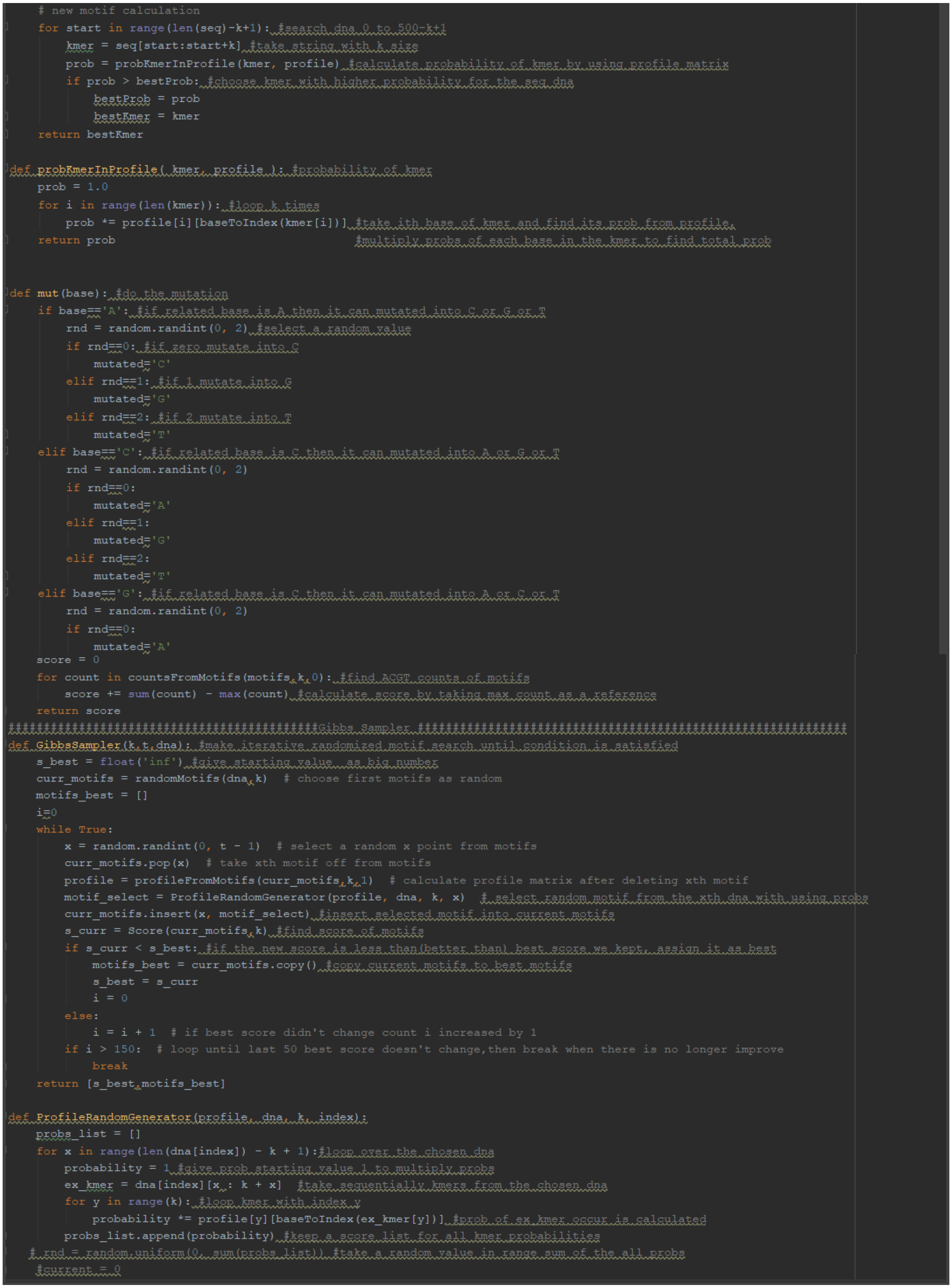
```
GAGAAACTTGTATCCGACCCCCGCAGCTTGCCAGCTCTTCGGGTATCATGGAGCCTATGGTTGAACGTGTCCGATAACGAACTTCGACATGATAAAGTCCC
TAGGAGTGAGCCTACCTCCGTTTGCGTCTTGTTACCAATGAAAAAGCTATGCACTTTGTACAGGGTGCCATCGGGTTTCTGAACTCTCAGATAGTGGGGAT
ATCGTCGAGGACCCCACAGTACCTCGGAGACCAAGTAGGGCACCCTATAGTTCGAAGCAGAACTATTTCGAGGGGCGAGCCCTCATCGTCTCTTCTGCGGA
TATCAAGGTTACTTTTTGGATACCTGAAACAAATCCCATGGTAGTCCTTAGACTTGGGAGTCTATCACCCCTAGGGCCCATATCTGGAAATAGACGCCAAG
CTCGTTCATCCCGTGGGACATCAAGCTTCCCCTTGATAAAGCACCCCGCTCGGGTATGGCAGAGAGGACGCCTTCTGAATTGTGCTATCCCTCGACCTTAT
TGTTTCGGGTAGCACCAGGAATCTGAACCATGTGAATGTGGACGTGGCGCGCGTACACCTTAATCTCCGGTTCATGCTAGGGATGTGGCTGCATGCTACGT
AGGGAATTATTTCGTAAGGCGAGCCCATACCGTCATTCATGCGGAAGAGTTAACACGATTGGAAGTAGGAATAGTTTCGAACCACGGTTACTAATCCTAAT
GTGCACGTAGGGTTGCACCGCACGCATGTGGAATTAGTGGCGAAGTACGATTCCACGACCGACGTACGATTCAACTATGCGGACGTGACGAGCTTCTTTTA
TGGCACTAAGTACGCCCTCCCCATTCTGGTAATCTTCATCCCTATCAGGGCTTGGAGTGAATGGTGAGGGTTATTCCCCAGGAACAGACTTCCTACTCACA
CATCAGTAATAGGGATAGTGCGAAAGCTCACTGACCACTGCCTATAGGGGGTGCTTACTTTTAGAAAAAGTGTCAGCCAGTATAACCCCACGAGGATTCGA
```

Windows (CRLF) Ln 10, Col 500 100%

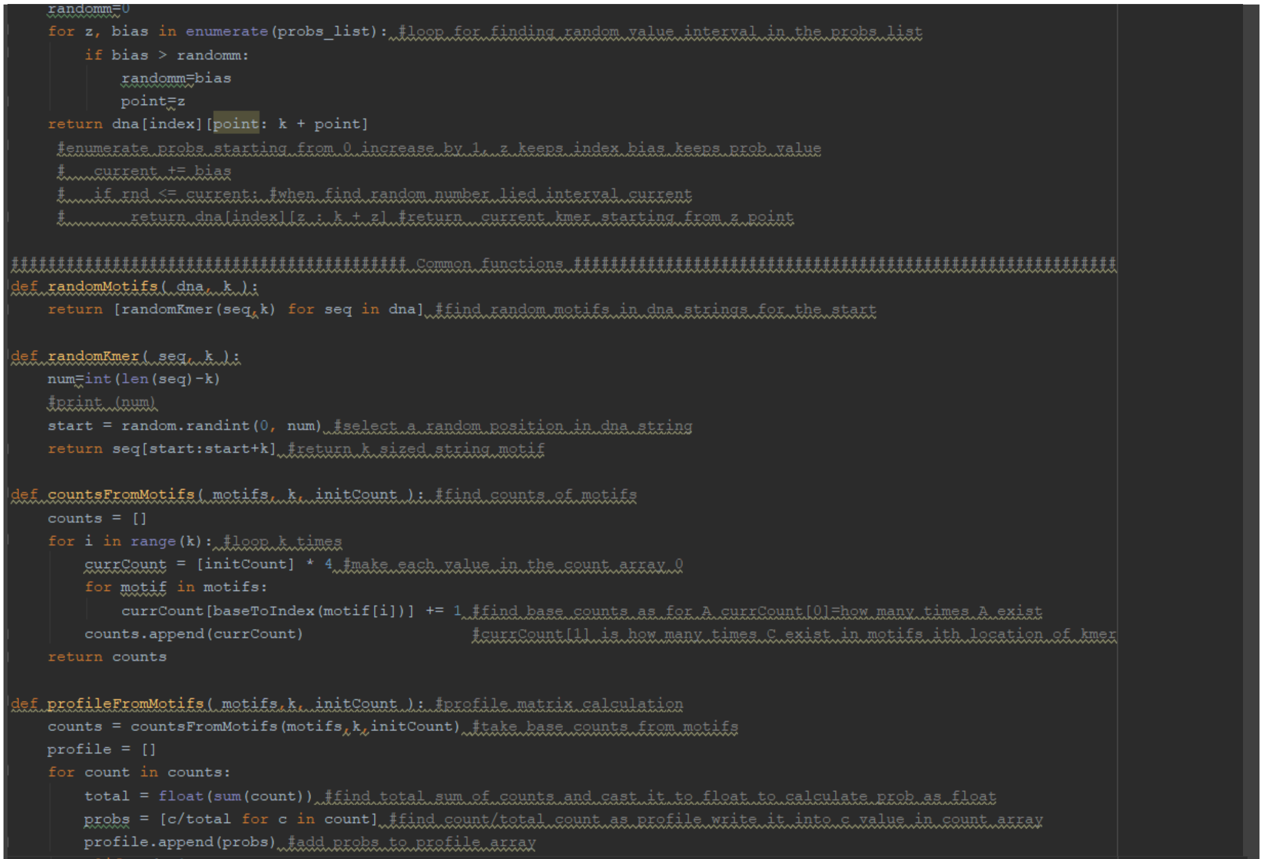
(Figure-3: Sample Input File with the total number of characters)



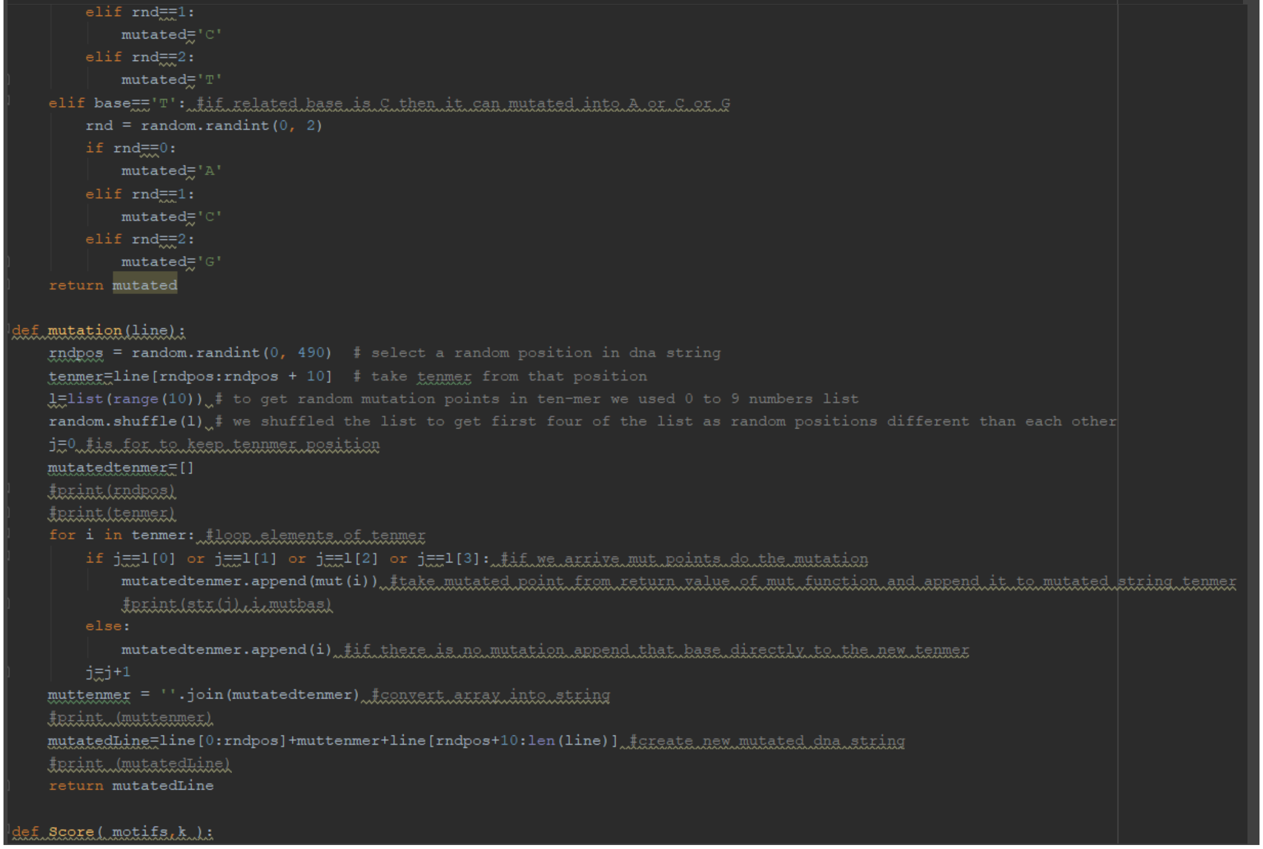
```python
import random

def indexToBase(i):
    return "ACGT"[i] #return a represent base for values 0->A, 1->C, 2->G, 3->T

def baseToIndex(a):
    return "ACGT".index(a) #return a represent value for bases A->0, C->1, G->2, T->3

def readInput( file, t ):
    dna = []
    i=0
    for i in range(t):
        line=file.readline().strip() #keep dna strings in dna array
        mutatedline=mutation(str(line))
        dna.append(mutatedline)
    return dna
###################################### Randomized Motif Search ###########################################################
def randomizedMotifSearch( k, dna ):
    bestMotifs = randomMotifs(dna, k)  #choose first motifs as random
    bestScore = Score(bestMotifs,k) #find best score of motifs
    while True:
        profile = profileFromMotifs(bestMotifs,k,1) #calulate profile matrix
        motifs = motifsFromProfile(dna,profile) #find motifs from dna strings by profile matrix
        score = Score(motifs,k) #find score of new motifs
        if score < bestScore: #keep lowest score as best score
            bestMotifs = motifs
            bestScore = score
        else:
            return bestMotifs #return best motifs

def motifsFromProfile( dna, profile ):
    return [bestKmerForProfile(seq, profile) for seq in dna] #send dna seq to find best motif for each seq

def bestKmerForProfile( seq, profile ):
    k = len(profile)
    bestProb = -1
    bestKmer = ''
```



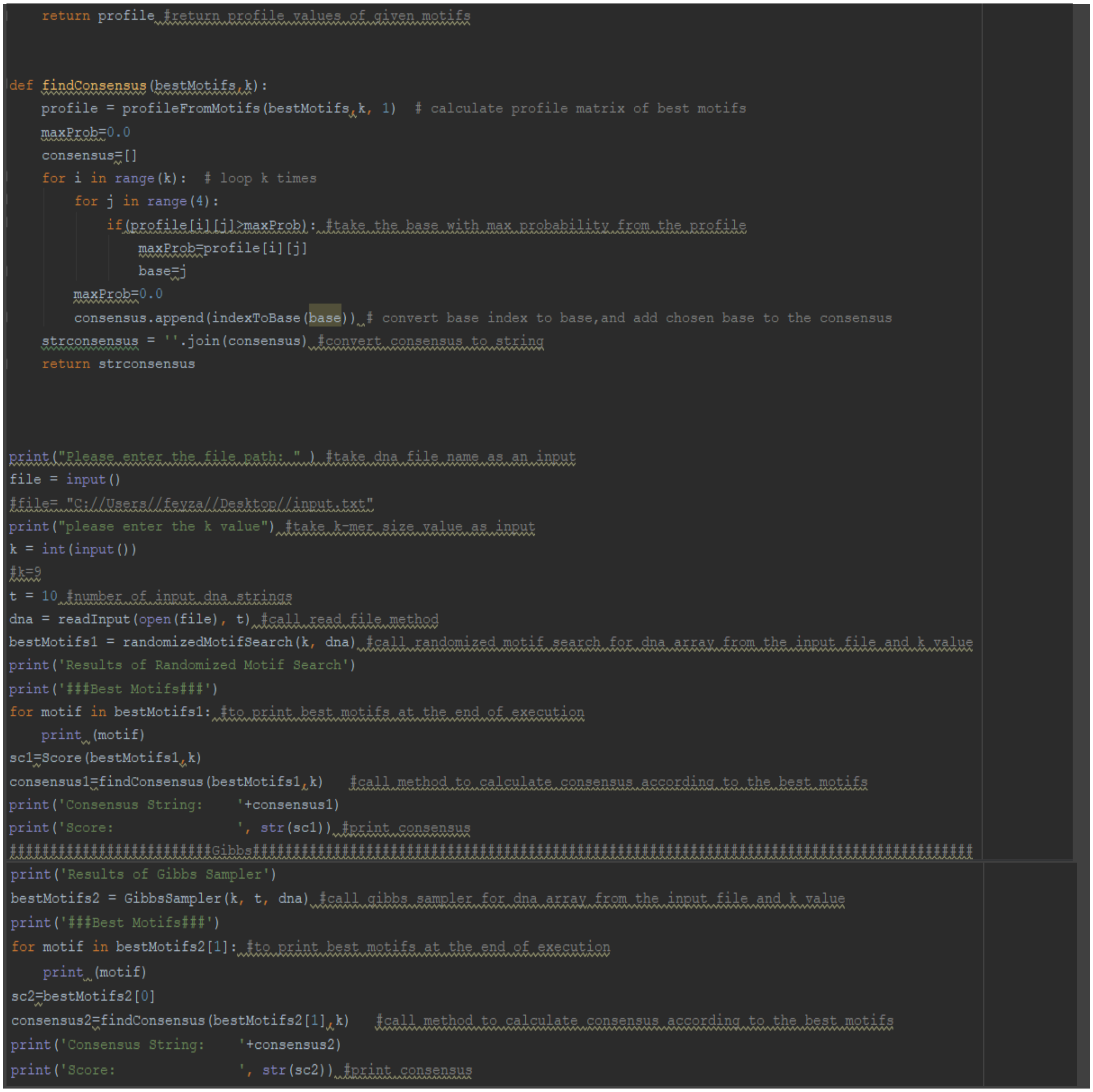
```
    # new motif calculation
    for start in range(len(seq)-k+1): #search dna 0 to 500-k+1
        kmer = seq[start:start+k] #take string with k size
        prob = probKmerInProfile(kmer, profile) #calculate probability of kmer by using profile matrix
        if prob > bestProb: #choose kmer with higher probability for the seq dna
            bestProb = prob
            bestKmer = kmer
    return bestKmer

def probKmerInProfile( kmer, profile ): #probability of kmer
    prob = 1.0
    for i in range(len(kmer)): #loop k times
        prob *= profile[i][baseToIndex(kmer[i])] #take ith base of kmer and find its prob from profile,
    return prob                                    #multiply probs of each base in the kmer to find total prob


def mut(base): #do the mutation
    if base=='A': #if related base is A then it can mutated into C or G or T
        rnd = random.randint(0, 2) #select a random value
        if rnd==0: #if zero mutate into C
            mutated='C'
        elif rnd==1: #if 1 mutate into G
            mutated='G'
        elif rnd==2: #if 2 mutate into T
            mutated='T'
    elif base=='C': #if related base is C then it can mutated into A or G or T
        rnd = random.randint(0, 2)
        if rnd==0:
            mutated='A'
        elif rnd==1:
            mutated='G'
        elif rnd==2:
            mutated='T'
    elif base=='G': #if related base is C then it can mutated into A or C or T
        rnd = random.randint(0, 2)
        if rnd==0:
            mutated='A'
    score = 0
    for count in countsFromMotifs(motifs,k,0): #find ACGT counts of motifs
        score += sum(count) - max(count) #calculate score by taking max count as a reference
    return score
############################################Gibbs Sampler #############################################################
def GibbsSampler(k,t,dna): #make iterative randomized motif search until condition is satisfied
    s_best = float('inf') #give starting value  as big number
    curr_motifs = randomMotifs(dna,k)  # choose first motifs as random
    motifs_best = []
    i=0
    while True:
        x = random.randint(0, t - 1)  # select a random x point from motifs
        curr_motifs.pop(x)  # take xth motif off from motifs
        profile = profileFromMotifs(curr_motifs,k,1)  # calculate profile matrix after deleting xth motif
        motif_select = ProfileRandomGenerator(profile, dna, k, x)  # select random motif from the xth dna with using probs
        curr_motifs.insert(x, motif_select) #insert selected motif into current motifs
        s_curr = Score(curr_motifs,k) #find score of motifs
        if s_curr < s_best: #if the new score is less than(better than) best score we kept, assign it as best
            motifs_best = curr_motifs.copy() #copy current motifs to best motifs
            s_best = s_curr
            i = 0
        else:
            i = i + 1  # if best score didn't change count i increased by 1
        if i > 150:  # loop until last 50 best score doesn't change,then break when there is no longer improve
            break
    return [s_best,motifs_best]

def ProfileRandomGenerator(profile, dna, k, index):
    probs_list = []
    for x in range(len(dna[index]) - k + 1):#loop over the chosen dna
        probability = 1 #give prob starting value 1 to multiply probs
        ex_kmer = dna[index][x : k + x]  #take sequentially kmers from the chosen dna
        for y in range(k): #loop kmer with index y
            probability *= profile[y][baseToIndex(ex_kmer[y])] #prob of ex kmer occur is calculated
        probs_list.append(probability) #keep a score list for all kmer probabilities
   # rnd = random.uniform(0, sum(probs_list)) #take a random value in range sum of the all probs
    #current = 0
```



```
    randomm=0
    for z, bias in enumerate(probs_list): #loop for finding random value interval in the probs list
        if bias > randomm:
            randomm=bias
            point=z
    return dna[index][point: k + point]
     #enumerate probs starting from 0 increase by 1, z keeps index bias keeps prob value
     #    current += bias
     #    if rnd <= current: #when find random number lied interval current
     #        return dna[index][z : k + z] #return  current kmer starting from z point

########################################### Common functions ############################################################
def randomMotifs( dna, k ):
    return [randomKmer(seq,k) for seq in dna] #find random motifs in dna strings for the start

def randomKmer( seq, k ):
    num=int(len(seq)-k)
    #print (num)
    start = random.randint(0, num) #select a random position in dna string
    return seq[start:start+k] #return k sized string motif

def countsFromMotifs( motifs, k, initCount ): #find counts of motifs
    counts = []
    for i in range(k): #loop k times
        currCount = [initCount] * 4 #make each value in the count array 0
        for motif in motifs:
            currCount[baseToIndex(motif[i])] += 1 #find base counts as for A currCount[0]=how many times A exist
        counts.append(currCount)                 #currCount[1] is how many times C exist in motifs ith location of kmer
    return counts

def profileFromMotifs( motifs,k, initCount ): #profile matrix calculation
    counts = countsFromMotifs(motifs,k,initCount) #take base counts from motifs
    profile = []
    for count in counts:
        total = float(sum(count)) #find total sum of counts and cast it to float to calculate prob as float
        probs = [c/total for c in count] #find count/total count as profile write it into c value in count array
        profile.append(probs) #add probs to profile array
```

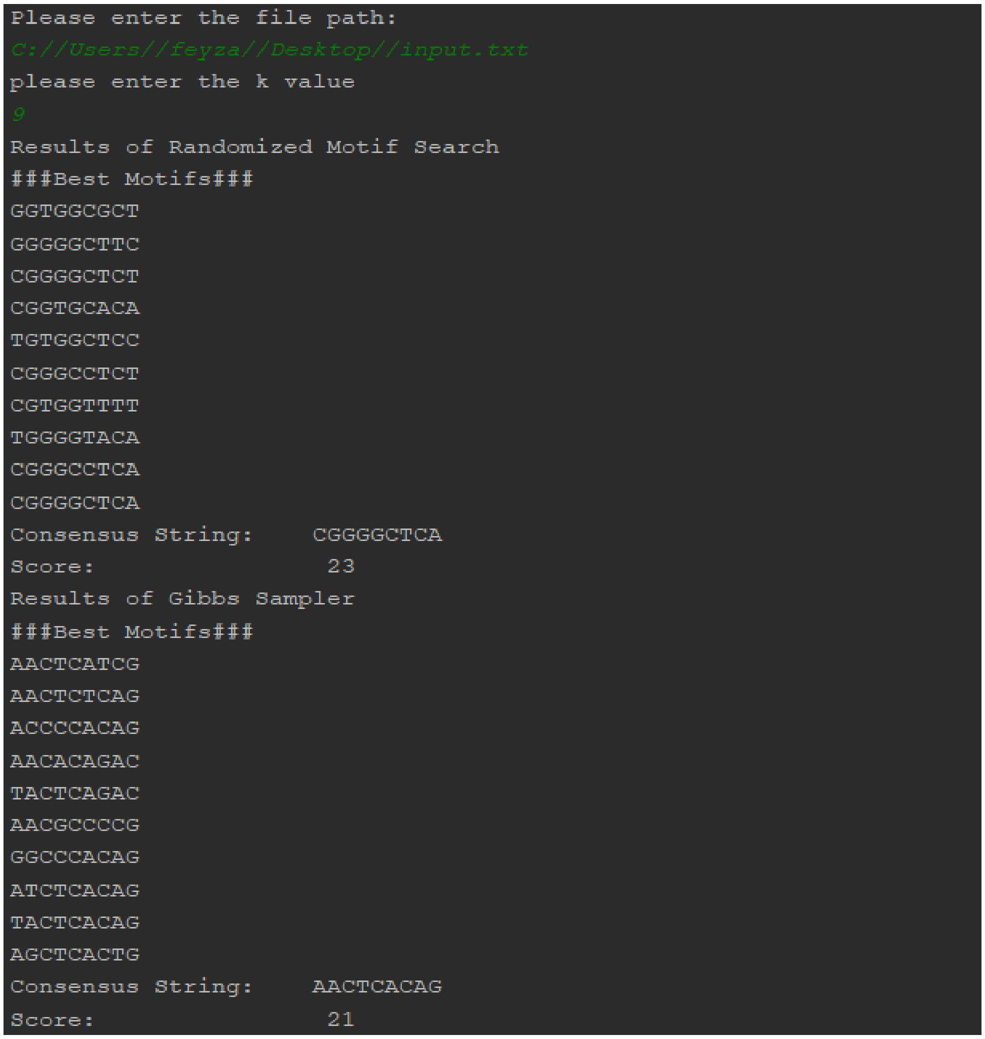


```
        elif rnd==1:
            mutated='C'
        elif rnd==2:
            mutated='T'
    elif base=='T': #if related base is C then it can mutated into A or C or G
        rnd = random.randint(0, 2)
        if rnd==0:
            mutated='A'
        elif rnd==1:
            mutated='C'
        elif rnd==2:
            mutated='G'
    return mutated

def mutation(line):
    rndpos = random.randint(0, 490)  # select a random position in dna string
    tenmer=line[rndpos:rndpos + 10]  # take tenmer from that position
    l=list(range(10)) # to get random mutation points in ten-mer we used 0 to 9 numbers list
    random.shuffle(l) # we shuffled the list to get first four of the list as random positions different than each other
    j=0 #is for to keep tennmer position
    mutatedtenmer=[]
    #print(rndpos)
    #print(tenmer)
    for i in tenmer: #loop elements of tenmer
        if j==l[0] or j==l[1] or j==l[2] or j==l[3]: #if we arrive mut points do the mutation
            mutatedtenmer.append(mut(i)) #take mutated point from return value of mut function and append it to mutated string tenmer
            #print(str(j),i,mutbas)
        else:
            mutatedtenmer.append(i) #if there is no mutation append that base directly to the new tenmer
        j=j+1
    muttenmer = ''.join(mutatedtenmer) #convert array into string
    #print (muttenmer)
    mutatedLine=line[0:rndpos]+muttenmer+line[rndpos+10:len(line)] #create new mutated dna string
    #print (mutatedLine)
    return mutatedLine

def Score( motifs,k ):
```



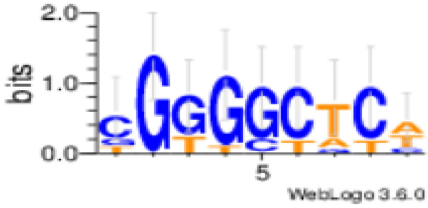
```
    return profile #return profile values of given motifs


def findConsensus(bestMotifs,k):
    profile = profileFromMotifs(bestMotifs,k, 1)  # calculate profile matrix of best motifs
    maxProb=0.0
    consensus=[]
    for i in range(k):  # loop k times
        for j in range(4):
            if(profile[i][j]>maxProb): #take the base with max probability from the profile
                maxProb=profile[i][j]
                base=j
        maxProb=0.0
        consensus.append(indexToBase(base)) # convert base index to base,and add chosen base to the consensus
    strconsensus = ''.join(consensus) #convert consensus to string
    return strconsensus



print("Please enter the file path: " ) #take dna file name as an input
file = input()
#file= "C://Users//feyza//Desktop//input.txt"
print("please enter the k value") #take k-mer size value as input
k = int(input())
#k=9
t = 10 #number of input dna strings
dna = readInput(open(file), t) #call read file method
bestMotifs1 = randomizedMotifSearch(k, dna) #call randomized motif search for dna array from the input file and k value
print('Results of Randomized Motif Search')
print('###Best Motifs###')
for motif in bestMotifs1: #to print best motifs at the end of execution
    print (motif)
sc1=Score(bestMotifs1,k)
consensus1=findConsensus(bestMotifs1,k)   #call method to calculate consensus according to the best motifs
print('Consensus String:    '+consensus1)
print('Score:               ', str(sc1)) #print consensus
#########################Gibbs##########################################################################################
print('Results of Gibbs Sampler')
bestMotifs2 = GibbsSampler(k, t, dna) #call gibbs sampler for dna array from the input file and k value
print('###Best Motifs###')
for motif in bestMotifs2[1]: #to print best motifs at the end of execution
    print (motif)
sc2=bestMotifs2[0]
consensus2=findConsensus(bestMotifs2[1],k)   #call method to calculate consensus according to the best motifs
print('Consensus String:    '+consensus2)
print('Score:               ', str(sc2)) #print consensus
```

(Figure-4: Codes of the Project)

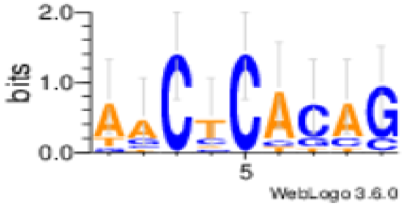


```
Please enter the file path:
C://Users//feyza//Desktop//input.txt
please enter the k value
9
Results of Randomized Motif Search
###Best Motifs###
GGTGGCGCT
GGGGGCTTC
CGGGGCTCT
CGGTGCACA
TGTGGCTCC
CGGGCCTCT
CGTGGTTTT
TGGGGTACA
CGGGCCTCA
CGGGGCTCA
Consensus String:     CGGGGCTCA
Score:                 23
Results of Gibbs Sampler
###Best Motifs###
AACTCATCG
AACTCTCAG
ACCCCACAG
AACACAGAC
TACTCAGAC
AACGCCCCG
GGCCCACAG
ATCTCACAG
TACTCACAG
AGCTCACTG
Consensus String:     AACTCACAG
Score:                 21
```

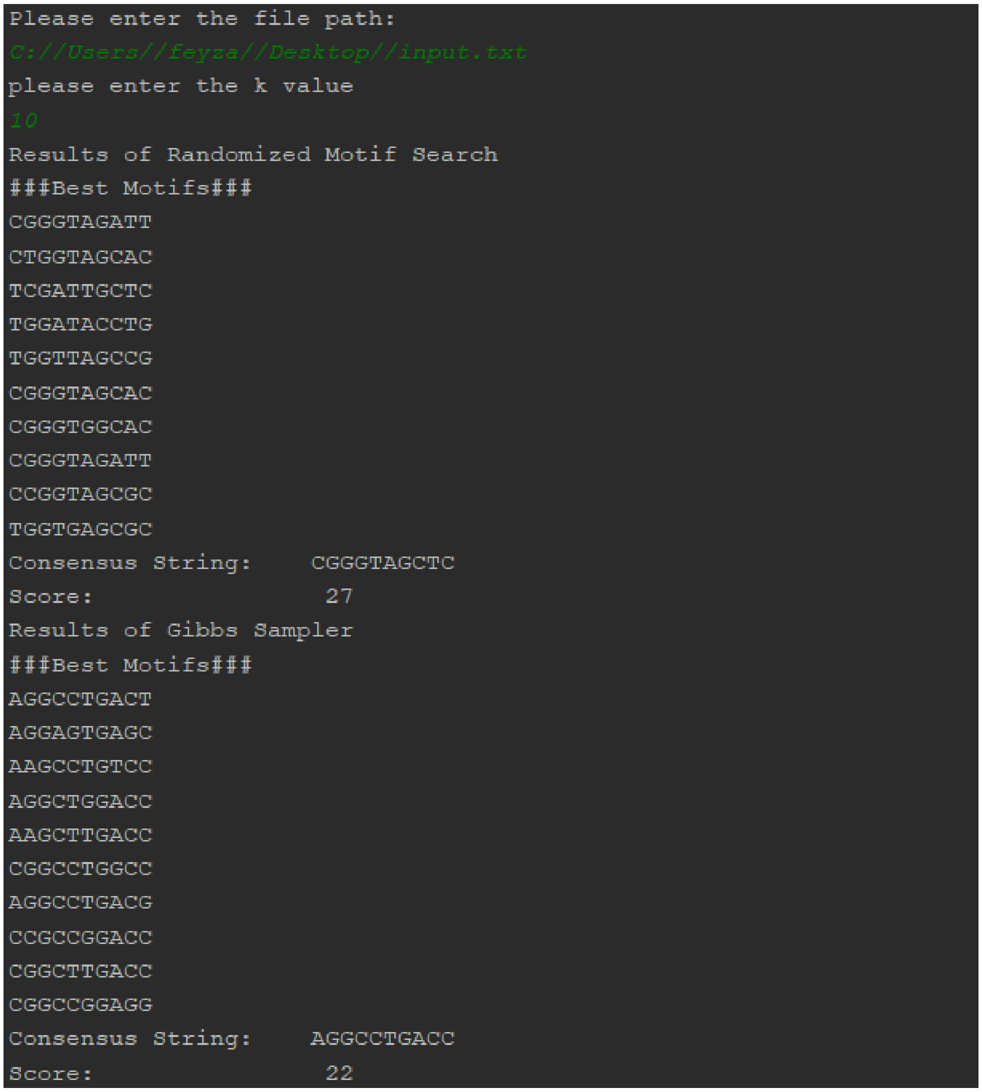
(Figure-5: Output of k is 9)



(Figure-6: Consensus string of Randomized Motif Search for k is 9)

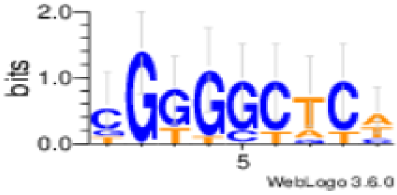


(Figure-7: Consensus string of Gibbs Sampler for k is 9)

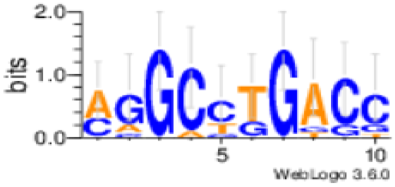


```
Please enter the file path:
C://Users//feyza//Desktop//input.txt
please enter the k value
10
Results of Randomized Motif Search
###Best Motifs###
CGGGTAGATT
CTGGTAGCAC
TCGATTGCTC
TGGATACCTG
TGGTTAGCCG
CGGGTAGCAC
CGGGTGGCAC
CGGGTAGATT
CCGGTAGCGC
TGGTGAGCGC
Consensus String:    CGGGTAGCTC
Score:                27
Results of Gibbs Sampler
###Best Motifs###
AGGCCTGACT
AGGAGTGAGC
AAGCCTGTCC
AGGCTGGACC
AAGCTTGACC
CGGCCTGGCC
AGGCCTGACG
CCGCCGGACC
CGGCTTGACC
CGGCCGGAGG
Consensus String:    AGGCCTGACC
Score:                22
```

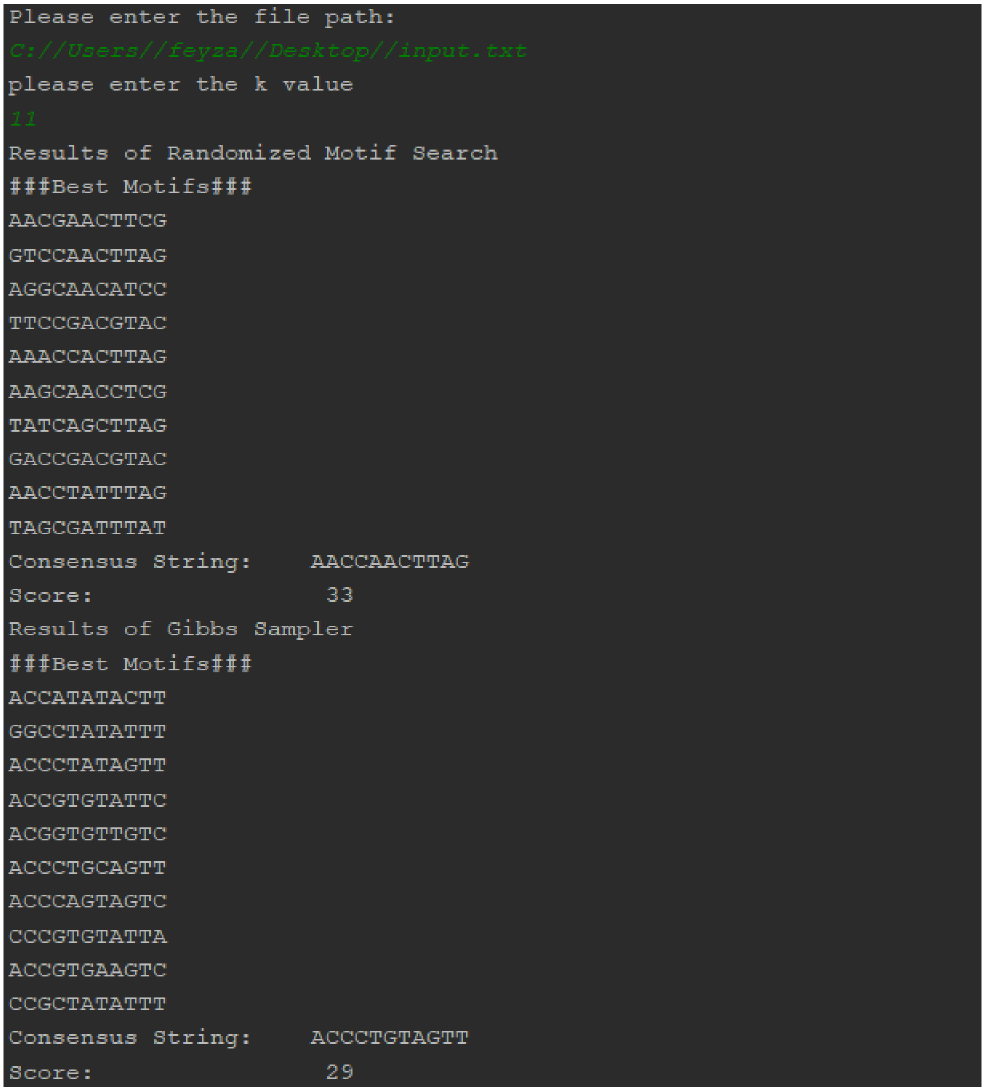
(Figure-8: Output of k is 10)



(Figure-9: Consensus string of Randomized Motif Search for k is 10)

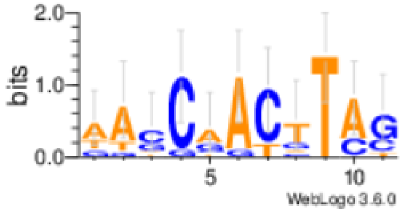


(Figure-10: Consensus string of Gibbs Sampler for k is 10)

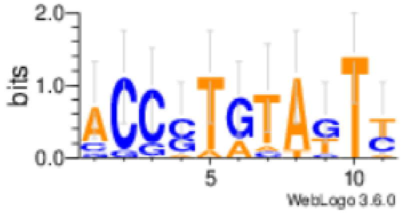


```
Please enter the file path:
C://Users//feyza//Desktop//input.txt
please enter the k value
11
Results of Randomized Motif Search
###Best Motifs###
AACGAACTTCG
GTCCAACTTAG
AGGCAACATCC
TTCCGACGTAC
AAACCACTTAG
AAGCAACCTCG
TATCAGCTTAG
GACCGACGTAC
AACCTATTTAG
TAGCGATTTAT
Consensus String:    AACCAACTTAG
Score:                33
Results of Gibbs Sampler
###Best Motifs###
ACCATATACTT
GGCCTATATTT
ACCCTATAGTT
ACCGTGTATTC
ACGGTGTTGTC
ACCCTGCAGTT
ACCCAGTAGTC
CCCGTGTATTA
ACCGTGAAGTC
CCGCTATATTT
Consensus String:    ACCCTGTAGTT
Score:                29
```

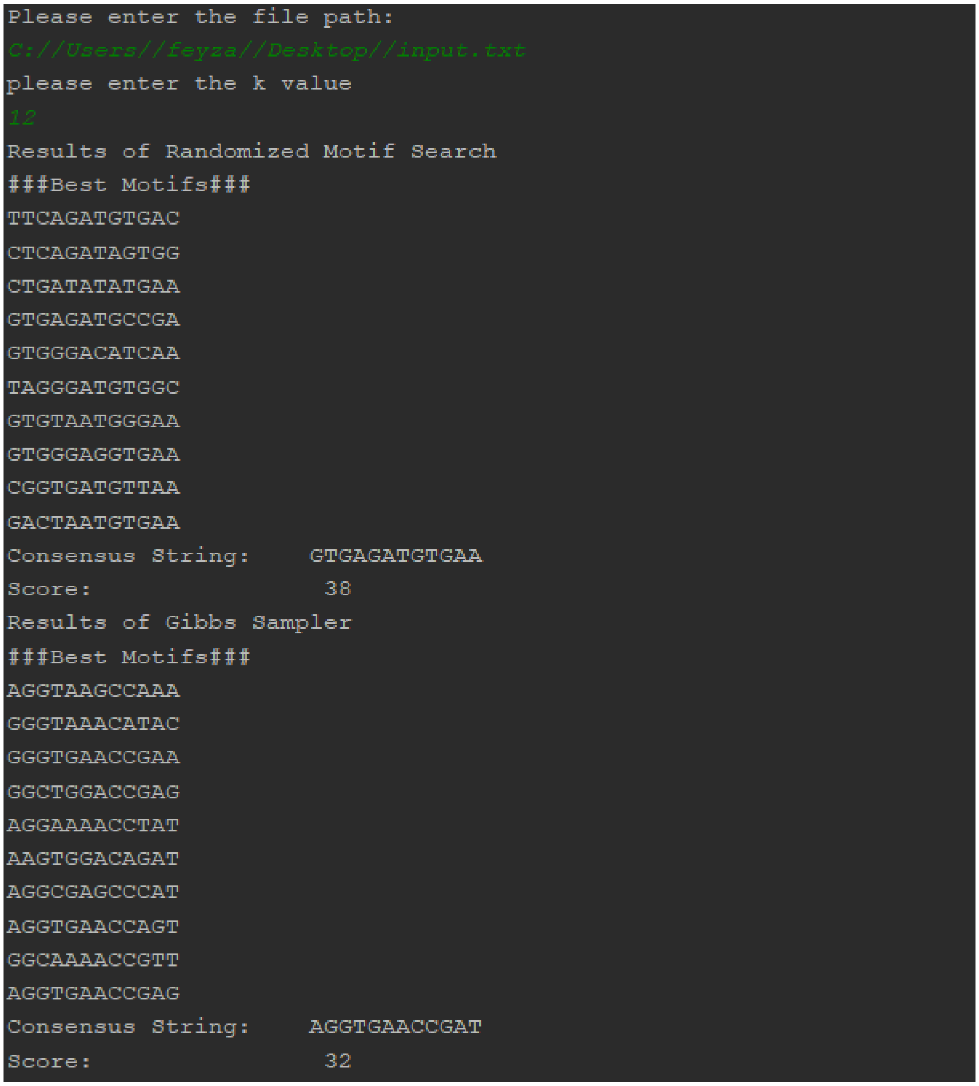
(Figure-11: Output of k is 11)



(Figure-12: Consensus string of Randomized Motif Search for k is 11)

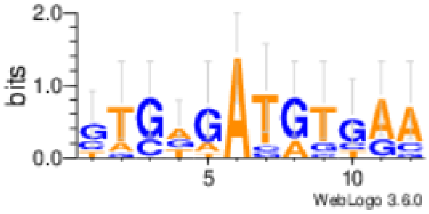


(Figure-13: Consensus string of Gibbs Sampler for k is 11)

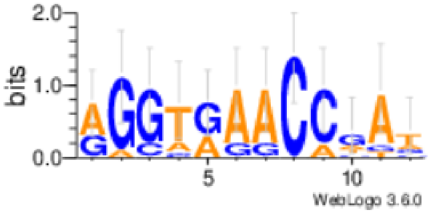


```
Please enter the file path:
C://Users//feyza//Desktop//input.txt
please enter the k value
12
Results of Randomized Motif Search
###Best Motifs###
TTCAGATGTGAC
CTCAGATAGTGG
CTGATATATGAA
GTGAGATGCCGA
GTGGGACATCAA
TAGGGATGTGGC
GTGTAATGGGAA
GTGGGAGGTGAA
CGGTGATGTTAA
GACTAATGTGAA
Consensus String:    GTGAGATGTGAA
Score:                 38
Results of Gibbs Sampler
###Best Motifs###
AGGTAAGCCAAA
GGGTAAACATAC
GGGTGAACCGAA
GGCTGGACCGAG
AGGAAAACCTAT
AAGTGGACAGAT
AGGCGAGCCCAT
AGGTGAACCAGT
GGCAAAACCGTT
AGGTGAACCGAG
Consensus String:    AGGTGAACCGAT
Score:                 32
```

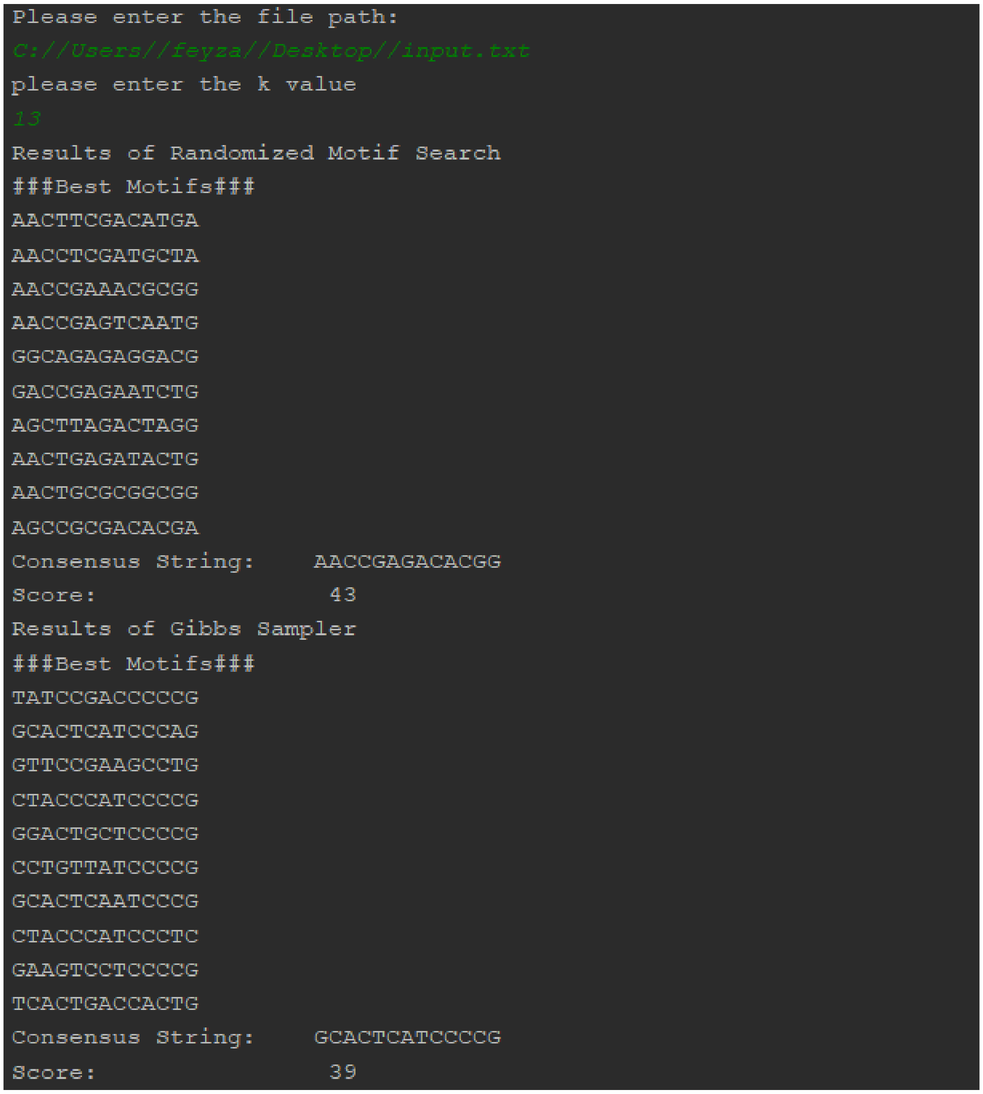
(Figure-14: Output of k is 12)



(Figure-15: Consensus string of Randomized Motif Search for k is 12)

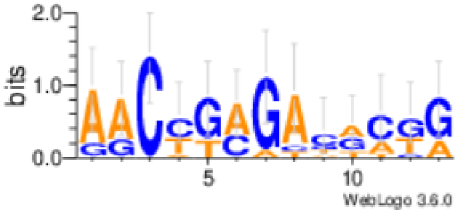


(Figure-16: Consensus string of Gibbs Sampler for k is 12)

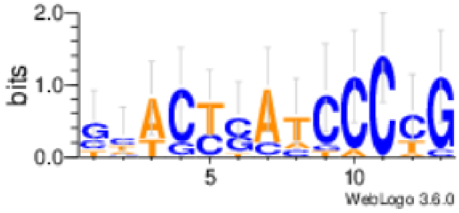


```
Please enter the file path:
C://Users//feyza//Desktop//input.txt
please enter the k value
13
Results of Randomized Motif Search
###Best Motifs###
AACTTCGACATGA
AACCTCGATGCTA
AACCGAAACGCGG
AACCGAGTCAATG
GGCAGAGAGGACG
GACCGAGAATCTG
AGCTTAGACTAGG
AACTGAGATACTG
AACTGCGCGGCGG
AGCCGCGACACGA
Consensus String:    AACCGAGACACGG
Score:                43
Results of Gibbs Sampler
###Best Motifs###
TATCCGACCCCCG
GCACTCATCCCAG
GTTCCGAAGCCTG
CTACCCATCCCCG
GGACTGCTCCCCG
CCTGTTATCCCCG
GCACTCAATCCCG
CTACCCATCCCTC
GAAGTCCTCCCCG
TCACTGACCACTG
Consensus String:    GCACTCATCCCCG
Score:                39
```

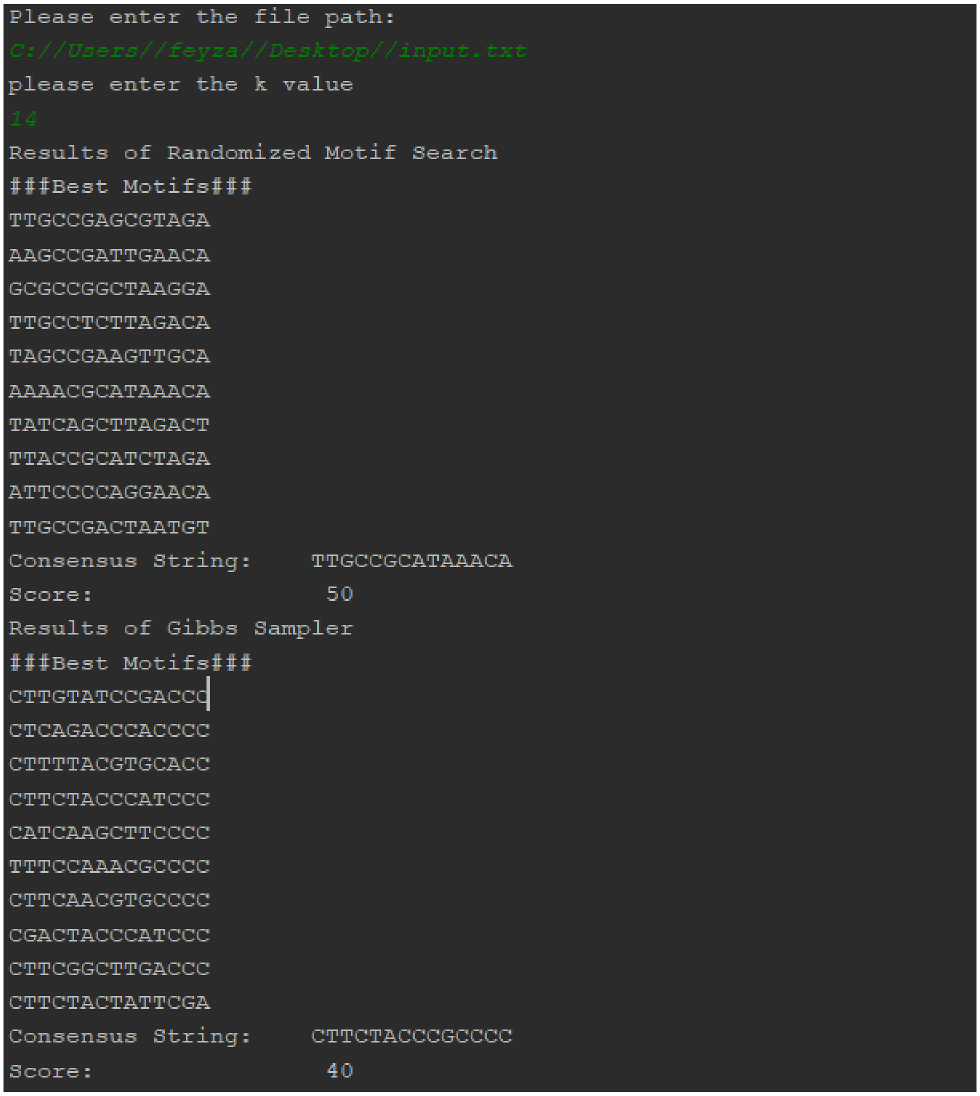
(Figure-17: Output of k is 13)



(Figure-18: Consensus string of Randomized Motif Search for k is 13)

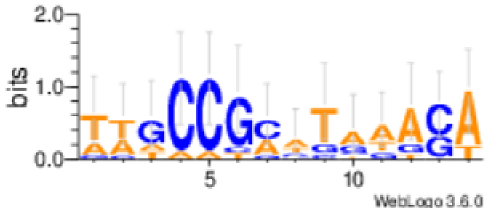


(Figure-19: Consensus string of Gibbs Sampler for k is 13)

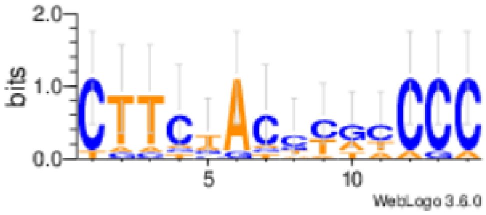


```
Please enter the file path:
C://Users//feyza//Desktop//input.txt
please enter the k value
14
Results of Randomized Motif Search
###Best Motifs###
TTGCCGAGCGTAGA
AAGCCGATTGAACA
GCGCCGGCTAAGGA
TTGCCTCTTAGACA
TAGCCGAAGTTGCA
AAAACGCATAAACA
TATCAGCTTAGACT
TTACCGCATCTAGA
ATTCCCCAGGAACA
TTGCCGACTAATGT
Consensus String:    TTGCCGCATAAACA
Score:                50
Results of Gibbs Sampler
###Best Motifs###
CTTGTATCCGACCC
CTCAGACCCACCCC
CTTTTACGTGCACC
CTTCTACCCATCCC
CATCAAGCTTCCCC
TTTCCAAACGCCCC
CTTCAACGTGCCCC
CGACTACCCATCCC
CTTCGGCTTGACCC
CTTCTACTATTCGA
Consensus String:    CTTCTACCCGCCCC
Score:                40
```

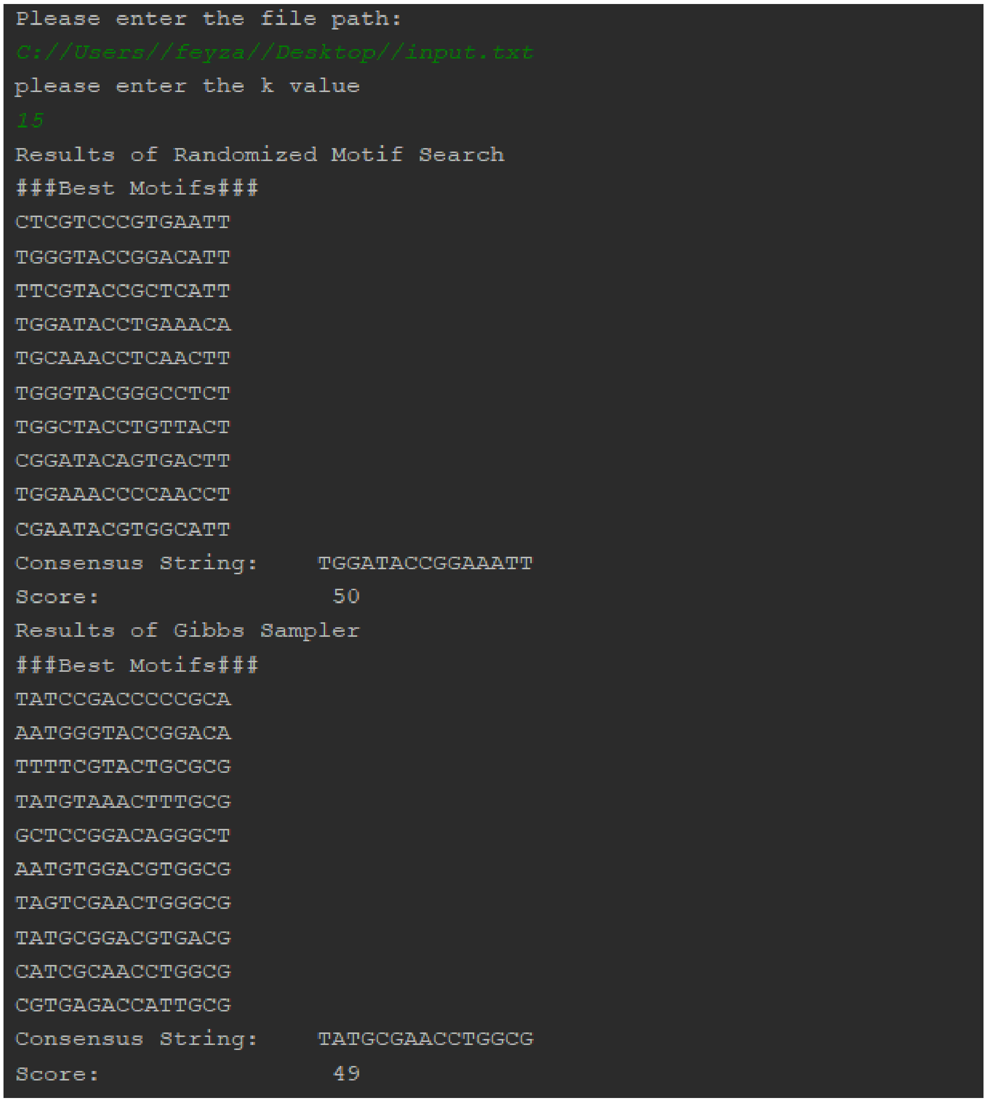
(Figure-20: Output of k is 14)



(Figure-21: Consensus string of Randomized Motif Search for k is 14)

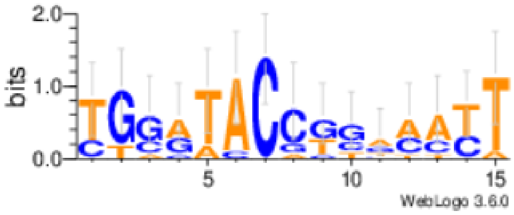


(Figure-22: Consensus string of Gibbs Sampler for k is 14)

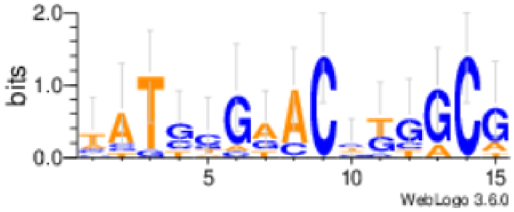


```
Please enter the file path:
C://Users//feyza//Desktop//input.txt
please enter the k value
15
Results of Randomized Motif Search
###Best Motifs###
CTCGTCCCGTGAATT
TGGGTACCGGACATT
TTCGTACCGCTCATT
TGGATACCTGAAACA
TGCAAACCTCAACTT
TGGGTACGGGCCTCT
TGGCTACCTGTTACT
CGGATACAGTGACTT
TGGAAACCCCAACCT
CGAATACGTGGCATT
Consensus String:    TGGATACCGGAAATT
Score:                50
Results of Gibbs Sampler
###Best Motifs###
TATCCGACCCCCGCA
AATGGGTACCGGACA
TTTTCGTACTGCGCG
TATGTAAACTTTGCG
GCTCCGGACAGGGCT
AATGTGGACGTGGCG
TAGTCGAACTGGGCG
TATGCGGACGTGACG
CATCGCAACCTGGCG
CGTGAGACCATTGCG
Consensus String:    TATGCGAACCTGGCG
Score:                49
```

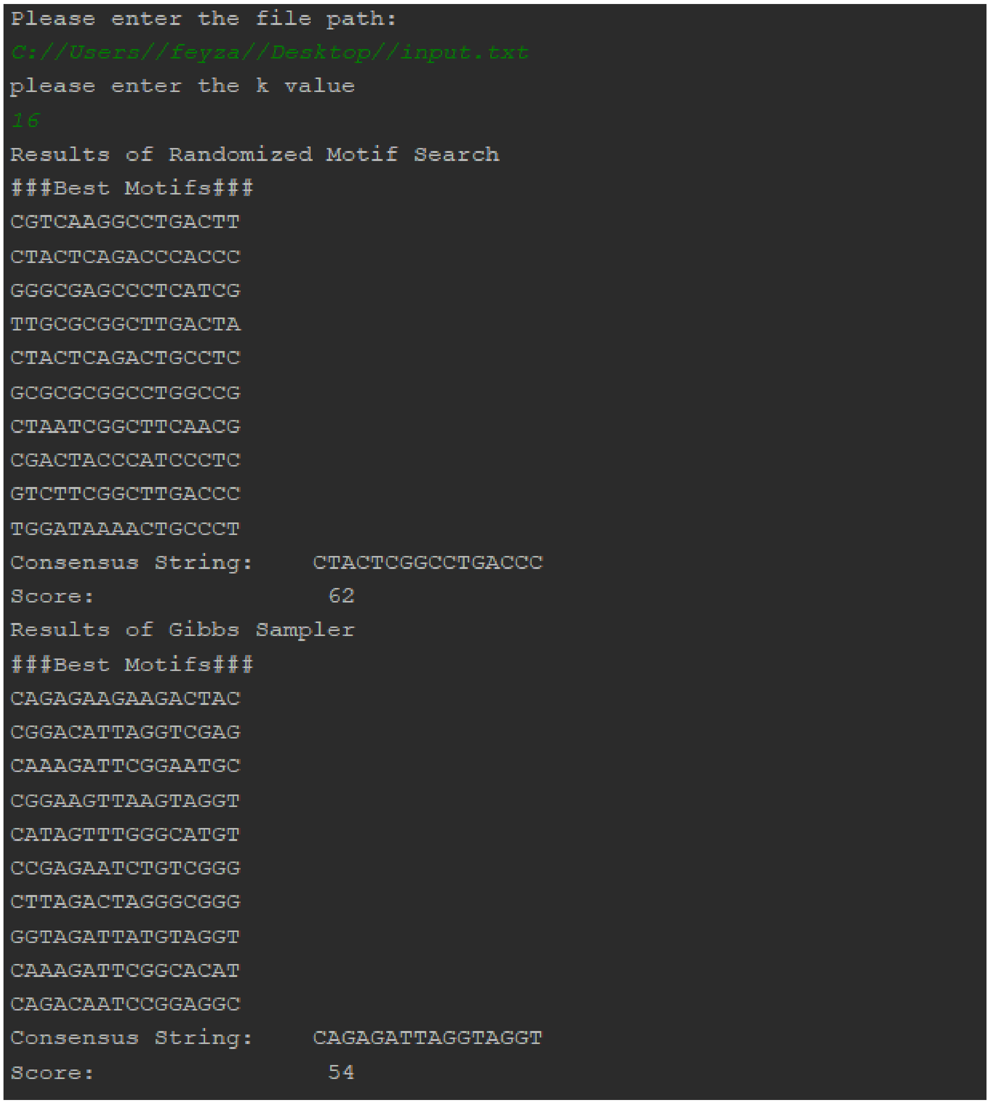
(Figure-23: Output of k is 15)



(Figure-24: Consensus string of Randomized Motif Search for k is 15)

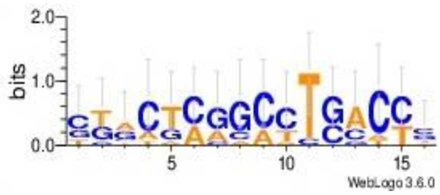


(Figure-25: Consensus string of Gibbs Sampler for k is 15)

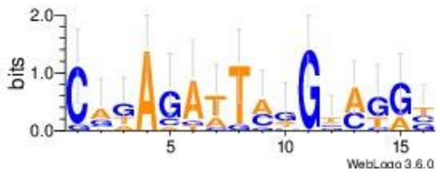


```
Please enter the file path:
C://Users//feyza//Desktop//input.txt
please enter the k value
16
Results of Randomized Motif Search
###Best Motifs###
CGTCAAGGCCTGACTT
CTACTCAGACCCACCC
GGGCGAGCCCTCATCG
TTGCGCGGCTTGACTA
CTACTCAGACTGCCTC
GCGCGCGGCCTGGCCG
CTAATCGGCTTCAACG
CGACTACCCATCCCTC
GTCTTCGGCTTGACCC
TGGATAAAACTGCCCT
Consensus String:    CTACTCGGCCTGACCC
Score:                 62
Results of Gibbs Sampler
###Best Motifs###
CAGAGAAGAAGACTAC
CGGACATTAGGTCGAG
CAAAGATTCGGAATGC
CGGAAGTTAAGTAGGT
CATAGTTTGGGCATGT
CCGAGAATCTGTCGGG
CTTAGACTAGGGCGGG
GGTAGATTATGTAGGT
CAAAGATTCGGCACAT
CAGACAATCCGGAGGC
Consensus String:    CAGAGATTAGGTAGGT
Score:                 54
```

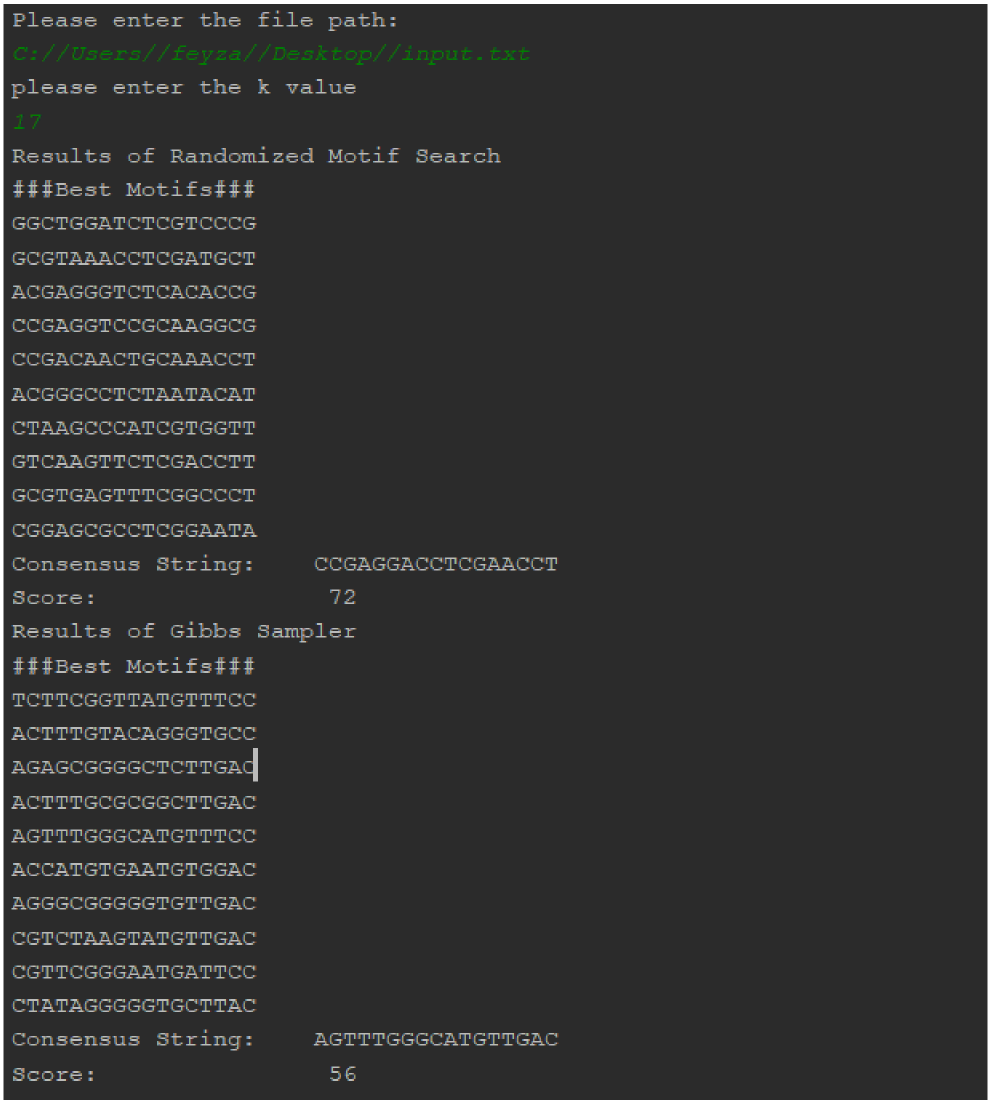
(Figure-26: Output of k is 16)



(Figure-27: Consensus string of Randomized Motif Search for k is 16)

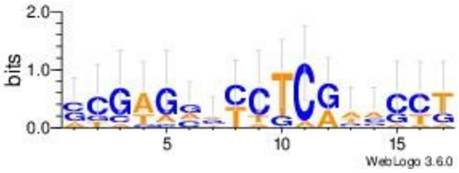


(Figure-28: Consensus string of Gibbs Sampler for k is 16)

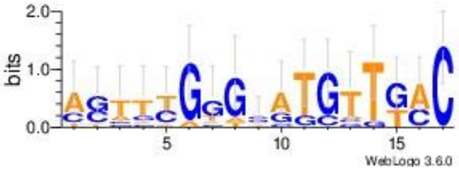


```
Please enter the file path:
C://Users//feyza//Desktop//input.txt
please enter the k value
17
Results of Randomized Motif Search
###Best Motifs###
GGCTGGATCTCGTCCCG
GCGTAAACCTCGATGCT
ACGAGGGTCTCACACCG
CCGAGGTCCGCAAGGCG
CCGACAACTGCAAACCT
ACGGGCCTCTAATACAT
CTAAGCCCATCGTGGTT
GTCAAGTTCTCGACCTT
GCGTGAGTTTCGGCCCT
CGGAGCGCCTCGGAATA
Consensus String:    CCGAGGACCTCGAACCT
Score:                72
Results of Gibbs Sampler
###Best Motifs###
TCTTCGGTTATGTTTCC
ACTTTGTACAGGGTGCC
AGAGCGGGGCTCTTGAC
ACTTTGCGCGGCTTGAC
AGTTTGGGCATGTTTCC
ACCATGTGAATGTGGAC
AGGGCGGGGGTGTTGAC
CGTCTAAGTATGTTGAC
CGTTCGGGAATGATTCC
CTATAGGGGGTGCTTAC
Consensus String:    AGTTTGGGCATGTTGAC
Score:                56
```

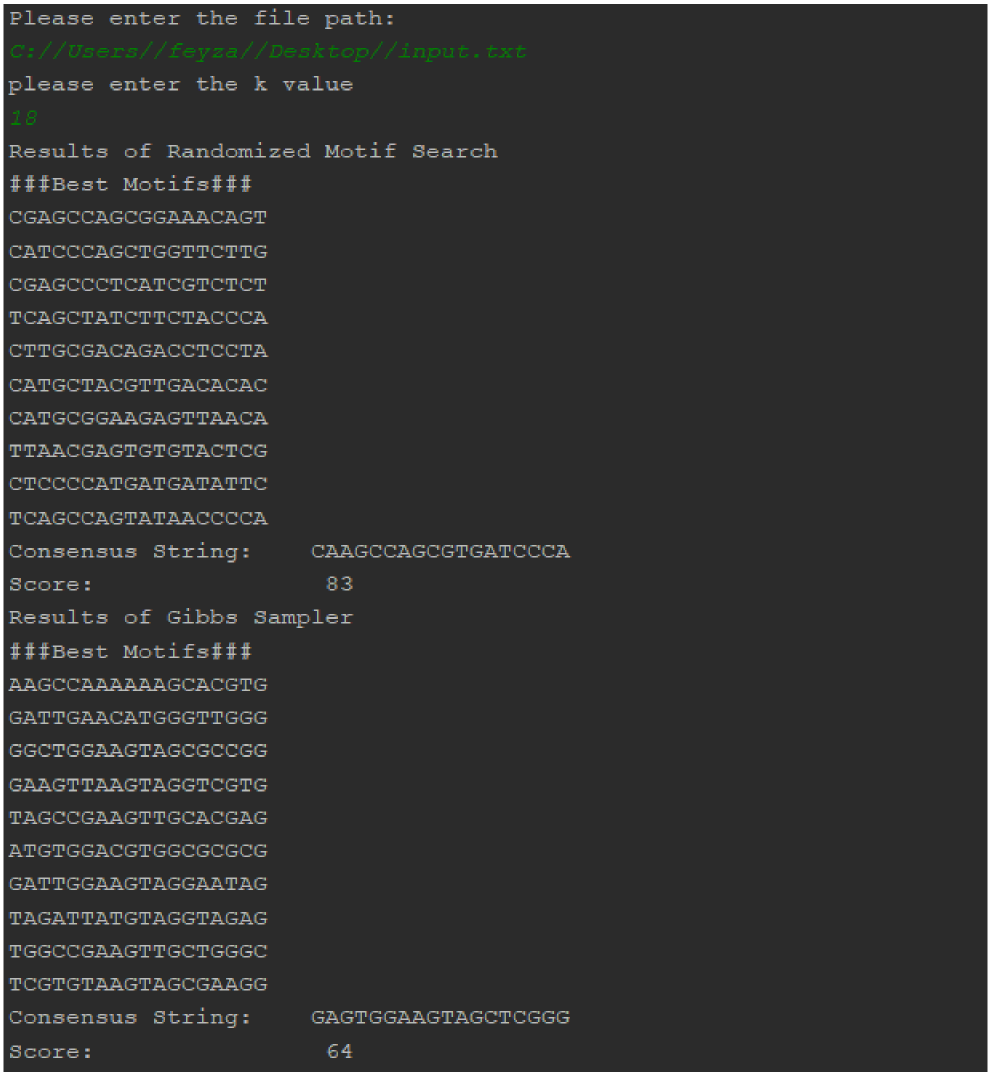
(Figure-29: Output of k is 17)



(Figure-30: Consensus string of Randomized Motif Search for k is 17)

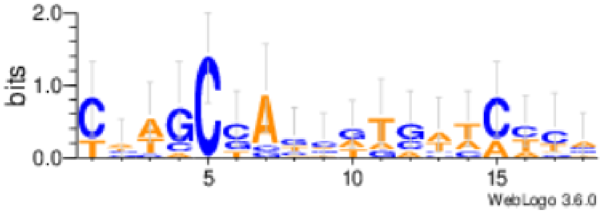


(Figure-31: Consensus string of Gibbs Sampler for k is 17)

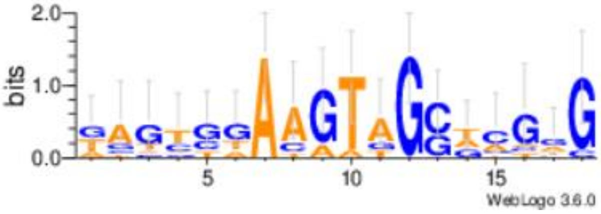


```
Please enter the file path:
C://Users//feyza//Desktop//input.txt
please enter the k value
18
Results of Randomized Motif Search
###Best Motifs###
CGAGCCAGCGGAAACAGT
CATCCCAGCTGGTTCTTG
CGAGCCCTCATCGTCTCT
TCAGCTATCTTCTACCCA
CTTGCGACAGACCTCCTA
CATGCTACGTTGACACAC
CATGCGGAAGAGTTAACA
TTAACGAGTGTGTACTCG
CTCCCCATGATGATATTC
TCAGCCAGTATAACCCCA
Consensus String:    CAAGCCAGCGTGATCCCA
Score:                83
Results of Gibbs Sampler
###Best Motifs###
AAGCCAAAAAAAGCACGTG
GATTGAACATGGGTTGGG
GGCTGGAAGTAGCGCCGG
GAAGTTAAGTAGGTCGTG
TAGCCGAAGTTGCACGAG
ATGTGGACGTGGCGCGCG
GATTGGAAGTAGGAATAG
TAGATTATGTAGGTAGAG
TGGCCGAAGTTGCTGGGC
TCGTGTAAGTAGCGAAGG
Consensus String:    GAGTGGAAGTAGCTCGGG
Score:                64
```

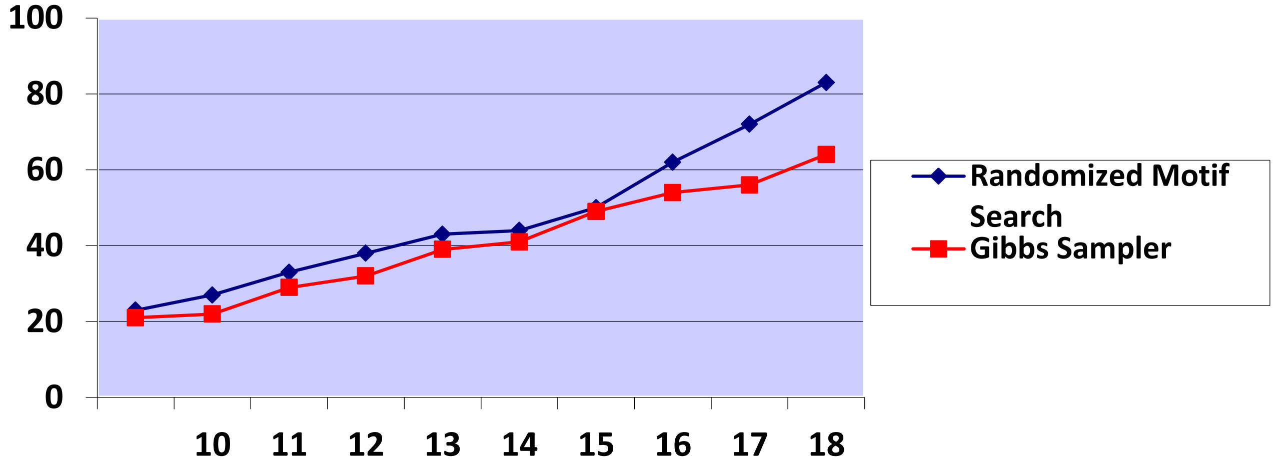
(Figure-32: Output of k is 18)



(Figure-33: Consensus string of Randomized Motif Search for k is 18)



(Figure-34: Consensus string of Gibbs Sampler for k is 18)



(Graph-1: Score of Randomized Motif Search and Gibbs Sampler for Different ‘k’ Values)

CONCLUSION

In this project, we used a txt file (input.txt) that has 10 lines, where each line contains strings of length 500 that include randomly generated genomic bases. Then, we selected a random position between 0 – 490 to insert the 10-mer that has 4 mutations in 4 random positions. In our programs, we take a k value which is the length of the consensus string.

In Randomized Motif Search, we chose a random motif with length k for each line. Then, we calculated the score with initial count zero for each base and the profile matrix of them. By choosing best motif, we found new motifs from DNA strings and calculated score and the profile matrix of new motifs. We repeated these processes until it believes that there is no more best score out there. It keeps best score value when one score comes higher than the best score algorithm halts and returns its best value so, it caught up by local optima. Finally, we had 10 motifs and found the consensus string using these.

In Gibbs Sampler, we chose a random motif with length k for each line and selected a random x point to take xth motif off from motifs. Then, we calculated the score with initial count one for each base and the profile matrix of them after deleting xth motif. By choosing a random motif from the xth DNA with using probabilities, we found new motifs from DNA strings and calculated score and the profile matrix of new motifs. We repeated these processes until the score not change in the last 50 iterations. Finally, we had 10 motifs and found the consensus string using these.

After, we implemented Randomize Motif Search and Gibbs Sampler, we get information that Gibbs Sampler randomly tries all possibilities and gives chance to all bases by taking initial count of bases, so it is better to see every chance and gives better score than Random Motif Search. However, Randomized Motif Search works faster than Gibbs Sampler because it returns best motif that it finds the first. Lastly, we noticed that the scores of Randomize Motif Search and Gibbs Sampler are mostly increasing as the k-mers' size increased.

At the end of the program, we have 10 motifs for each algorithm. According to these motifs' count probability, we found the consensus string and checked them by using “http://weblogo.threeplusone.com/create.cgi” link. Then, we added all outputs and consensus strings in our report.

REFERENCES

http://bix.ucsd.edu/bioalgorithms/presentations/Ch12_RandAlgs.pdf

https://www.cs.helsinki.fi/u/tpkarkka/teach/16-17/MAiB/MAiB2016-lecture2.pdf

http://www.mit.edu/~ilkery/papers/GibbsSampling.pdf