Question 1

Write a program to find the anti-complimentary sequence of an input DNA sequence

```
def complement(seq):
  seqr=""
  for i in range(len(seq)):
    if seq[i] == "A":
      seqr=seqr+"T"
    if seq[i] == "T":
      seqr=seqr+"A"
    if seq[i] == "G":
      seqr=seqr+"C"
    if seq[i] == "C":
      seqr=seqr+"G"
  return seqr


def reverse(seq):
  return seq[::-1]


def first():
  print("Give the input sequence : ")
  seq=input()
  print("Complement of the given sequence: "+complement(seq))
  print("Anti complement/ reverse complement of given sequence: "+reverse(complemen


first()
```

```
Give the input sequence :
AAGGAAAC
Complement of the given sequence: TTCCTTTG
Anti complement/ reverse complement of given sequence: GTTTCCTT
```

Question 2

Intrinsic terminators have two prominent structural features:

(1) a sequence of nucleotides that includes an inverted repeat (i.e., the sequence 5'-CGGATG|CATCCG-3' contains an inverted repeat centered at the "|" because "5'-CGGATG-3'" reads "5'-CATCCG-3'" on its complementary strand), and

(2) a run of roughly six uracils immediately following the inverted repeat. Write a function to identify intrinsic terminators from the sequence.

```
def finding(seq):
  req="UUUUUU"
  if (seq.find(req) != -1):
    seq=seq[: seq.find(req)]
    return seq
  else:
    return False


def finds(x,seq):
  result=[]
  for i in range(len(seq)):
    if seq[i]==x:
      result.append(i)
  return result


def truth(x,y,seq):
  if (seq[x]=="T" and seq[y]=="A") or (seq[x]=="A" and seq[y]=="T") or(seq[x]=="C"
    return True
  else:
    return False


def cond(seq):
  n=len(seq)
  if seq[n-1]=="A":
    index=finds("T",seq)
  elif seq[n-1]=="T":
    index=finds("A",seq)
  elif seq[n-1]=="C":
    index=finds("G",seq)
  elif seq[n-1]=="G":
    index=finds("C",seq)
  #print(index)
  for i in index:
    if (n-1-i)%2!=0:
      #print(i)
      x=i
      y=n-1

      while(x<y):
        cond=0
        if truth(x,y,seq):
          #print(x,y)
          x=x+1
          y=y-1
        else:
          cond=1
          x=y+1
      if cond==0:
        #print("here i am")
        seq=seq[i:x]
        return seq
        break
```

```
  return -1



#fixing length by 6
def second():
  print("Give the sequence: ")
  seq=input()
  sub_seq=finding(seq)
  #print(sub_seq)
  #print(len(sub_seq))
  if sub_seq:
    x=cond(sub_seq)
    if x==-1:
      print("No intrinsic terminator")
    else:
      print("The intrinsic terminator is: "+x)
  else:
    print("No intrinsic terminator")



print("How many times do you want to verify for intrinsic terminator: ")
n=int(input())
for i in range(n):
  second()
```

```
How many times do you want to verify for intrinsic terminator:
3
Give the sequence:
AAGGAAACGTTT
No intrinsic terminator
Give the sequence:
AAGGAAACGTTTUUUUUU
The intrinsic terminator is: AAAC
Give the sequence:
AAGGAAACTUUUUUU
No intrinsic terminator
```

Question 3

Given DNA sequence, arrange the codons in the decreasing order of frquency.

```
def identify_codons(seq):
  codons=[]
  while (len(seq)>2):
    codons.append(seq[:3])
    seq=seq[3:]
  return codons
```

```
def codons_x(codons):
  Codon_count= {}
  for element in codons:
    if element in Codon_count:
      Codon_count[element] += 1
    else:
      Codon_count[element] = 1
  Codon_list= sorted(Codon_count.items(),key=operator.itemgetter(1),reverse=True)
  return Codon_list
  #for key, value in Codon_list.items():
   # print(key, ' : ', value)

import operator
def main():
  print("Give the nucleotide sequence ")
  seq=input()
  print("The codons identified in the sequence are: ")
  codons= identify_codons(seq)
  codons=codons_x(codons)
  print("(codon,frequency)")
  for i in codons:
    print(i)



main()
```

```
Give the nucleotide sequence
ACTACTGGG
The codons identified in the sequence are:
(codon,frequency)
('ACT', 2)
('GGG', 1)
```

Question 4

Given two strings s and t of equal length, the Hamming distance between s and t, denoted dH(s,t), is the number of corresponding symbols that differ in s and t. Write a function to compute dH(s,t)

```
def distance(seq1,seq2):
  res=0
  for i in range(len(seq1)):
    if seq1[i]!=seq2[i]:
      res+=1
  return res


def hamming():
  print("Give the first sequence ")
  seq1=input()
```

```
  print("Give the second sequence ")
  seq2=input()
  print("The hamming distance is "+ str(distance(seq1,seq2)))


hamming()
```

```
    Give the first sequence
    AGCT
    Give the second sequence
    ATCG
    The hamming distance is 2
```

Question 5

Given Two protein strings s and t in FASTA format (each of length at most 100 aa), write a program to find the optimal global alignment score (use match score, mismatch and gap penalties as variables)

```
def needlmen(seq1,seq2):
  gap=-1
  mismatch=-1
  match=1
  n= len(seq1)
  m= len(seq2)
  res = [ [ 0 for i in range(m+1) ] for j in range(n+1) ]
  #Initialization
  count=0
  for i in range(n+1):
    res[i][0]=count
    count-=1
  count=0
  for i in range(m+1):
    res[0][i]=count
    count-=1
  print(res)
  for i in range(1,n+1):
    for j in range(1,m+1):
      x=res[i-1][j]+gap
      y=res[i][j-1]+gap
      if seq1[i-1]==seq2[j-1]:
        z=res[i-1][j-1]+match
      else:
        z=res[i-1][j-1]+mismatch
      res[i][j]=max(x,y,z)
  print(res)
  return res[n][m]


print(needlmen("ACTG","ACTG"))
```

```
[[0, -1, -2, -3, -4], [-1, 0, 0, 0, 0], [-2, 0, 0, 0, 0], [-3, 0, 0, 0, 0], [-
[[0, -1, -2, -3, -4], [-1, 1, 0, -1, -2], [-2, 0, 2, 1, 0], [-3, -1, 1, 3, 2],
4
```

```
def alignment():
  print("Give the sequence 1: ")
  seq1=input()
  print("Give the sequence 2: ")
  seq2=input()
  print("The optimal global alignment score is: "+str(needlmen(seq1,seq2)))


alignment()
```

```
Give the sequence 1:
gactt
Give the sequence 2:
att
[[0, -1, -2, -3], [-1, 0, 0, 0], [-2, 0, 0, 0], [-3, 0, 0, 0], [-4, 0, 0, 0],
[[0, -1, -2, -3], [-1, -1, -2, -3], [-2, 0, -1, -2], [-3, -1, -1, -2], [-4, -2
The optimal global alignment score is: 1
```