Open Source SW

" Homework#2 "

담당교수	황두성 교수님
학번	32164959
이름	허전진

Problem 1

A DNA sequence is a string made up of the letters A, T, G, and C. To find the complement of a DNA sequence, As are replaced by Ts, Ts by As, Gs by Cs, and Cs by Gs. For example, the complement of AATTGCCGT is TTAACGGCA.

In [25]:

```
def dna_short(old):
    trans_table = str.maketrans('tagc','atcg') #지정한 문자를 특정 문자로 변환
    new = old.translate(trans_table)
    return new

def dna_long(old):
    new = "" #if case로 전체를 분리하여 변환
    for _ in range(0, len(old)):
        if(old[_] == 't'):
            new += 'a'
        elif(old[_] == 'a'):
            new += 't'
        elif(old[_] == 'g'):
            new += 'c'
        elif(old[_] == 'c'):
            new += 'g'
        else:
            new += old[_]
    return new

# Test Case
p53="""1 ttcccatcaa gccctagggc tcctcgtggc tgctgggagt tgtagtctga acgcttctat
61 cttggcgaga agcgcctacg ctccccctac cgagtcccgc ggtaattctt aaagcacctg
121 caccgccccc ccgccgcctg cagagggcgc agcaggtctt gcacctcttc tgcatctcat
181 tctccaggct tcagacctgt ctccctcatt caaaaaatat ttattatcga gctcttactt
241 gctacccagc actgatatag gcactcagga atacaacaat gaataagata gtagaaaaat
301 tctatatcct cataaggctt acgtttccat gtactgaaag caatgaacaa ataaatctta
361 tcagagtgat aagggttgtg Waaggagatta aataagatgg tgtgatataa agtatctggg
421 agaaaacgtt agggtgtgat attacggaaa gccttcctaa aaaatgacat tttaactgat
481 gagaagaaag gatccagctg agagcaaacg caaaagcttt cttccttcca cccttcatat
541 ttgacacaat gcaggattcc tccaaaatga tttccaccaa ttctgccctc acagctctgg
601 cttgcagaat tttccacccc aaaatgttag tatctacggc accaggtcgg cgagaatcct
661 gactctgcac cctcctcccc aactccattt cctttgcttc ctccggcagg cggattactt
721 gcccttactt gtcatggcga ctgtccagct ttgtgccagg agcctcgcag ggttgatgg
781 gattggggtt ttcccctccc atgtgctcaa gactggcgct aaaagttttg agcttctcaa
"""

print(dna_short(p53))
print(dna_long(p53))
```

```
1 aagggtagtt cgggatcccg aggagcaccg acgaccctca acatcagact tgcgaagata
61 gaaccgctct tcgcggatgc gagggggatg gctcagggcg ccattaagaa tttcgtggac
121 gtggcggggg ggcggcggac gtctcccgcg tcgtccagaa cgtggagaag acgtagagta
181 agaggtccga agtctggaca gagggagtaa gttttttata aataatagct cgagaatgaa
241 cgatgggtcg tgactatatc cgtgagtcct tatgttgtta cttattctat catcttttta
301 agatatagga gtattccgaa tgcaaaggta catgactttc gttacttgtt tatttagaat
361 agtctcacta ttcccaacac Wttcctctaat ttattctacc acactatatt tcatagaccc
421 tcttttgcaa tcccacacta taatgccttt cggaaggatt ttttactgta aaattgacta
481 ctcttctttc ctaggtcgac tctcgtttgc gttttcgaaa gaaggaaggt gggaagtata
541 aactgtgtta cgtcctaagg aggttttact aaaggtggtt aagacgggag tgtcgagacc
601 gaacgtctta aaaggtgggg ttttacaatc atagatgccg tggtccagcc gctcttagga
661 ctgagacgtg ggaggagggg ttgaggtaaa ggaaacgaag gaggccgtcc gcctaatgaa
721 cgggaatgaa cagtaccgct gacaggtcga aacacggtcc tcggagcgtc cccaactacc
781 ctaaccccaa aaggggaggg tacacgagtt ctgaccgcga ttttcaaaac tcgaagagtt

1 aagggtagtt cgggatcccg aggagcaccg acgaccctca acatcagact tgcgaagata
61 gaaccgctct tcgcggatgc gagggggatg gctcagggcg ccattaagaa tttcgtggac
121 gtggcggggg ggcggcggac gtctcccgcg tcgtccagaa cgtggagaag acgtagagta
181 agaggtccga agtctggaca gagggagtaa gttttttata aataatagct cgagaatgaa
241 cgatgggtcg tgactatatc cgtgagtcct tatgttgtta cttattctat catcttttta
301 agatatagga gtattccgaa tgcaaaggta catgactttc gttacttgtt tatttagaat
361 agtctcacta ttcccaacac Wttcctctaat ttattctacc acactatatt tcatagaccc
421 tcttttgcaa tcccacacta taatgccttt cggaaggatt ttttactgta aaattgacta
```

```
481 ctcttctttc ctaggtcgac tctcgtttgc gttttcgaaa gaaggaaggt gggaagtata
541 aactgtgtta cgtcctaagg aggttttact aaaggtggtt aagacgggag tgtcgagacc
601 gaacgtctta aaaggtgggg ttttacaatc atagatgccg tggtccagcc gctcttagga
661 ctgagacgtg ggaggagggg ttgaggtaaa ggaaacgaag gaggccgtcc gcctaatgaa
721 cgggaatgaa cagtaccgct gacaggtcga aacacggtcc tcggagcgtc cccaactacc
781 ctaaccccaa aaggggaggg tacacgagtt ctgaccgcga ttttcaaaac tcgaagagtt
```

Problem 2

Develop a function that finds the minimum or maximum value in a list, depending on the caller's request.

1. **Write a loop (including initialization) to find both the minimum value in a list and that value's index in one pass through the list.**
2. **Write a function named min index that takes a list and returns a tuple containing the minimum value in the list and that value's index in the list.**
3. **Write a function named max index that takes a list and returns a tuple containing the maximum value in the list and that value's index in the list.**

In [7]:

```
def min_index(nums): #using loop to find min
    min=nums[0]
    min_idx=0
    for i in range(0,len(nums)):
        if(nums[i]<min):
            min=nums[i]
            min_idx=i
    return min, min_idx

def max_index(nums):#using loop to find max
    max=nums[0]
    max_idx=0
    for i in range(0,len(nums)):
        if(nums[i]>max):
            max=nums[i]
            max_idx=i
    return max, max_idx

def min_index_short(n): #using method
    return min(n), n.index(min(n))

def max_index_short(n):
    return max(n), n.index(max(n))

nums=[7,3,4,2,9,8,10,1,6,5]

print("min : %d, idx : %d"%min_index(nums))
print("max : %d, idx : %d"%max_index(nums))
print("-----to short-----")
print("min : %d, idx : %d"%min_index(nums))
print("max : %d, idx : %d"%max_index(nums))
```

```
min : 1, idx : 7
max : 10, idx : 6
-----to short-----
min : 1, idx : 7
max : 10, idx : 6
```

Problem 3

Design and implement a class Country that stores the information on countries such as nation name, capital city, population, and area. Then write a program that reads in a set of countries and prints

In [10]:

```python
class Country:
    def __init__(self, name, area, pop, den, cap):
        self.name=name
        self.area=area
        self.pop=pop
        self.den=den
        self.cap=cap

def largest_area(c):
    target = []
    for _ in range(0, len(c)):
        target.append(c[_].area)
    return c[target.index(max(target))].name

def largest_pop(c):
    target = []
    for _ in range(0, len(c)):
        target.append(c[_].pop)
    return c[target.index(max(target))].name

def largest_den(c):
    target = []
    for _ in range(0, len(c)):
        target.append(c[_].den)
    return c[target.index(max(target))].name

def capital(c):
    cap = []
    for _ in range(0, len(c)):
        if(c[_].cap):
            cap.append(c[_].cap)
    return cap

korea = Country('Korea',1003,5178,509,'Seoul')
usa = Country('USA',98315,33100,35,'Washington')
china = Country('China',96000,143932,148,'Beijing')

print(largest_area([korea,usa,china]))
print(largest_pop([korea,usa,china]))
print(largest_den([korea,usa,china]))
print(capital([korea,usa,china]))
```

```
USA
China
Korea
['Seoul', 'Washington', 'Beijing']
```

Problem 4

Based on object-oriented programming, design and implement each class for geometry objects on the next page.

1. **Implement and test the class on each object**
2. **Place those classes into a geometry module. Then write a program that prints a result for the chosen object depending on a user's values.**

In [4]:

```python
from math import pi

class geometry_area:
    def rectangle(self, a, b):
        return a*b

    def circle(self, r):
        return r ** 2 * pi
    # and other methods..

class geometry_perimeter:
    def rectangle(self, a, b):
        return 2 * (a+b)
    def circle(self, r):
        return r * 2 * pi
    #and other methods..

class geometry_volume:
    def rectanglar_box(self, a, b, c):
        return a*b*c
    def sphere(self, r):
        return r ** 3 * 4/3 * pi

g_a = geometry_area()
g_p = geometry_perimeter()
g_v = geometry_volume()

print("3 * 4 rectangle area : %f" % g_a.rectangle(3,4))
print("3 circle perimeter : %f" % g_p.circle(3))
print("2,3,4 rectanglar volume : %f" % g_v.rectanglar_box(2,3,4))
```

```
3 * 4 rectangle area : 12.000000
3 circle perimeter : 18.849556
2,3,4 rectanglar volume : 24.000000
```

Problem 5

Design a class Msg that models an e-mail message. A message has a recipient, a sender, and a message text. Support the following methods:

• A constructor that takes the sender and recipient
• A method append that appends a line of text to the message body
• A method str that returns the whole string like this:

In [5]:

```
class Msg:
    msg_content = ""
    def __init__(self, f, t):
        self.msg_from = f
        self.msg_to=t

    def writeMsg(self, c):
        self.msg_content=c

    def __str__(self):
        return "From:%s \nTo: %s\nContent: %s"%(self.msg_from, self.msg_to, self.msg_content)

new_msg = Msg("Heo", "Hwang")
new_msg.writeMsg("Thank you for your service")
print(str(new_msg))
```

```
From:Heo
To: Hwang
Content: Thank you for your service
```

Problem 6

– *Makefile*

```
CC= gcc
CFLAG= -I.
EXE= MatrixCalc
$(EXE): subtraction.o multiplication.o division.o main.o
	$(CC) -o $(EXE) subtraction.o multiplication.o division.o main.o
subtraction.o: subtraction.c subtraction.h
	$(CC) -c subtraction.c $(CFLAG)
multiplication.o: multiplication.c multiplication.h
	$(CC) -c multiplication.c $(CFLAG)
division.o: division.c division.h
	$(CC) -c division.c $(CFLAG)
main.o: main.c subtraction.h multiplication.h division.h
	$(CC) -c main.c $(CFLAG)
clean:
	rm -f *.o $(EXE)
```

– *Execution*

```
root@os:/home/ubuntu/OpenSourcePython/Report2/Problem6# ls
division.c  main.c    multiplication.c  subtraction.c
division.h  Makefile  multiplication.h  subtraction.h
root@os:/home/ubuntu/OpenSourcePython/Report2/Problem6# make
gcc -c subtraction.c -l.
gcc -c multiplication.c -l.
gcc -c division.c -l.
gcc -c main.c -l.
gcc -o MatrixCalc subtraction.o multiplication.o division.o main.o
root@os:/home/ubuntu/OpenSourcePython/Report2/Problem6# ./MatrixCalc
A's row : 2
A's column : 2
>> input Matrix A
>> 2 4
>> 3 5

B's row : 2
B's column : 2
>> input Matrix B
>> 3 3
>> 8 6

1. Subtraction / 2. Multiplication / 3.Element-wise Division / 0. Quit
>> 1
result
-1      1
-5      -1
1. Subtraction / 2. Multiplication / 3.Element-wise Division / 0. Quit
>> 2
result
38      30
49      39
1. Subtraction / 2. Multiplication / 3.Element-wise Division / 0. Quit
>> 3
result
0       1
0       0
1. Subtraction / 2. Multiplication / 3.Element-wise Division / 0. Quit
>> 0
```

Problem 7

Design and implement a class Letter for authoring a simple letter. In the constructor, supply the names of the sender and the recipient

In [5]:

```python
class Letter:
    text = ""
    def __init__(self, letterfrom, letterto):
        self.l_from = letterfrom
        self.l_to = letterto
    def addLine(self, line):
        self.text += "\n" + line
    def get_text(self):
        return "Dear " + self.l_to + ":\n" + self.text + "\n\nSincerely,\n" + self.l_from

letter = Letter("Heo", "Hwang")
letter.addLine("first line of the body")
letter.addLine("second, line of the body")
letter.addLine("third,  line of the body")
letter.addLine("last,  line of the body")
print(letter.get_text())
```

```
Dear Hwang:

first, line of the body
second, line of the body
third,  line of the body
last,  line of the body

Sincerely,
Heo
```

Problem 8

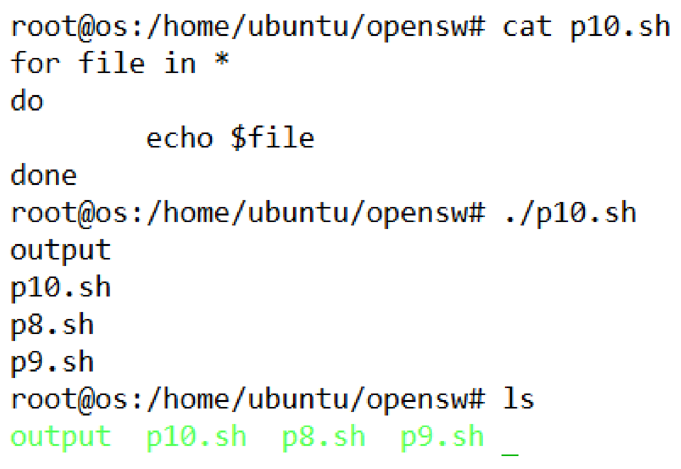
```
root@os:/home/ubuntu/opensw# cat p8.sh
fun(){
        arr=$1
        echo "The size : ${#arr[*]}"
        echo "The array : ${arr[*]}"
}

arr=(1 2 3 4 5 6 7)
fun ${arr[*]}
root@os:/home/ubuntu/opensw# ./p8.sh
The size : 7
The array : 1 2 3 4 5 6 7
```

Problem 9

– Loop count: 22

```
root@os:/home/ubuntu/opensw# cat p9.sh
for((v1 = 12; v1 < 34; v1++))
do
        echo "$v1"
done > output
root@os:/home/ubuntu/opensw# ./p9.sh
root@os:/home/ubuntu/opensw# ls
output  p10.sh  p8.sh  p9.sh
root@os:/home/ubuntu/opensw# cat output
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
```

Problem 10



```
root@os:/home/ubuntu/opensw# cat p10.sh
for file in *
do
        echo $file
done
root@os:/home/ubuntu/opensw# ./p10.sh
output
p10.sh
p8.sh
p9.sh
root@os:/home/ubuntu/opensw# ls
output  p10.sh  p8.sh  p9.sh _
```