opensource report#2

June 20, 2020

Problem 1

[34]:
```
# pseudo code
# def DNA(dna):
#   declare the new object to store new string
#   for i in range(0,length of string):
#       if(dna[i] is 't'): add 'a' to new string and continue to increment
#       if(dna[i] is 'a'): add 't' to new string and continue to increment
#       if(dna[i] is 'c'): add 'g' to new string and continue to increment
#       if(dna[i] is 'g'): add 'c' to new string and continue to increment
#       if(dna[i] is space): add just space to new string
#                            and continue to increment
#       if(dna[i] is not 't','a','c','g' and space): just add dna[i]
#                                                    and continue to increment
#       return the new string


def DNA(dna):
    chgd=""
    for i in range(0,len(dna)):
        if(dna[i]=='t'):
            chgd = chgd+'a'
            continue
        if(dna[i]=='a'):
            chgd = chgd+'t'
            continue
        if(dna[i]=='c'):
            chgd = chgd+'g'
            continue
        if(dna[i]=='g'):
            chgd = chgd+'c'
            continue
        if(dna[i]==' '):
            chgd = chgd+' '
            continue
        if(dna[i] != 't','a','c','g',' '):
            chgd = chgd+dna[i]
            continue
```

```
    return chgd


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 aaggagatta 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 gggttgatgg
781 gattggggtt ttcccctccc atgtgctcaa gactggcgct aaaagttttg agcttctcaa
"""
print(DNA(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 gtttttata aataatagct cgagaatgaa
241 cgatgggtcg tgactatatc cgtgagtcct tatgttgtta cttattctat catcttttta
301 agatatagga gtattccgaa tgcaaaggta catgactttc gttacttgtt tatttagaat
361 agtctcacta ttcccaacac ttcctctaat 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

[2]:
```
def min_index(nums):
    min=nums[0]
    minx=0
    for i in range(0,len(nums)):
        if(nums[i]<min):
            min=nums[i]
            minx=i
    return min, minx
```

```
def max_index(nums):
    max=nums[0]
    maxx=0
    for i in range(0,len(nums)):
        if(nums[i]>max):
            max=nums[i]
            maxx=i
    return max, maxx


nums=[7,3,4,2,9,8,10,1,6,5]
nums_min, nums_min_index = min_index(nums)
nums_max, nums_max_index = max_index(nums)
print("min : %d, min_index : %d"%(nums_min, nums_min_index))
print("max : %d, max_index : %d"%(nums_max, nums_max_index))
```

```
min : 1, min_index : 7
max : 10, max_index : 6
```

Problem 3

[19]:
```
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 max_area(lst, n):
    max = lst[0].area
    for i in range(0,n):
        if(lst[i].area>max):
            max=lst[i].area
    for i in range(0,n):
        if(lst[i].area==max):
            return lst[i].name

def max_pop(lst, n):
    max = lst[0].pop
    for i in range(0,n):
        if(lst[i].area>max):
            max=lst[i].pop
    for i in range(0,n):
        if(lst[i].pop==max):
            return lst[i].name

def max_den(lst, n):
```

```
    max = lst[0].den
    for i in range(0,n):
        if(lst[i].den>max):
            max=lst[i].area
    for i in range(0,n):
        if(lst[i].den==max):
            return lst[i].name

def exist_cap(lst, n):
    conlst=[]
    for i in range(0, n):
        if(lst[i].cap != None):
            conlst.append(lst[i].name)
    return conlst


korea = Country('korea',1003,5178,509,'Seoul')
usa = Country('usa',98315,33100,35,'Washington')
china = Country('china',96000,143932,148,'Beijing')
countries = [korea,usa,china]
print("The Country which has largest area : %s"%max_area(countries, 3))
print("The Country which has largest population : %s"%max_pop(countries, 3))
print("The Country which has largest population density : %s"
      %max_den(countries, 3))
print("The Country which has capital city : %s"%exist_cap(countries, 3))
```

```
The Country which has largest area : usa
The Country which has largest population : china
The Country which has largest population density : korea
The Country which has capital city : ['korea', 'usa', 'china']
```

Problem 4

[17]:
```
from math import pi, sqrt

class geometry_area:
    def square(self, s):
        """get A of square with s"""
        A = s**2
        return A

    def rectangle(self, a, b):
        """get A of rectangle with s"""
        A = a*b
        return A

    def circle(self, r):
        """get A of circle with r"""
```

```
        A = pi * r**2
        return A

    def triangle(self, b, h):
        """get A of triangle with b, h"""
        A = 0.5*b*h
        return A

    def parallelogram(self, b, h):
        """get A of parallelogram with b,h"""
        A = b*h
        return A

    def circle_ring(self, R, r):
        """get A of circle_ring with R,r"""
        A = pi * (R**2 - r**2)
        return A

    def trapezoid(self, h, a, b):
        """get A of trapezoid with h,a,b"""
        A = h * (a+b) / 2
        return A

    def rectangular_box(self, a, b, c):
        """get A of rectangular_box with a,b,c"""
        A = 2*a*b + 2*b*c + 2*a*c
        return A

    def cube(self, l):
        """get A of cube with l"""
        A=6 * (l**2)
        return A

    def cylinder(self, r, h):
        """get A of cylinder with r,h"""
        A = 2 * pi * r * (r+h)
        return A


    def right_circular_cone(self, r, s):
        """get A of right_circular_cone with r,s"""
        A = pi * (r**2) + math.pi * r * s
        return A

    def sphere(self, r):
        """get S of sphere with r"""
        S = 4 * pi * (r**2)
```

```
        return S

class geometry_busbar:
    def right_circular_cone(self, r, h):
        """get s of right_circular_cone with r,h"""
        s = sqrt((r**2) + (h**2))
        return s

class geometry_perimeter:
    def square(self, s):
        '''get P of square with s'''
        return 4*s

    def parallelogram(self, a, b):
        '''get P of parrallelogram with a, b'''
        return 2*a + 2*b

    def circle(self, r):
        '''get P of circle with r'''
        return 2*pi*r

    def triangle(self, a, b, c):
        '''get P of triangle with a,b,c'''
        return a+b+c

    def rectangle(self, a, b):
        '''get P of rentangle with a,b'''
        return 2*(a+b)

    def trapezoid(self, a, b, c, d):
        '''get P of trapezoid with a,b,c,d'''
        return a+b+c+d;

    def circular_sector(self, r, seta):
        '''get P(length) of circular sector with r, seta'''
        return r * seta

class geometry_pythagorean:
    def pythagorean_theorem(self, a, b):
        """get c of pythagorean_theorem with a,b"""
        c = sqrt((a**2) + (b**2))
        return c

class geometry_volume:
    def sphere(self, r):
        '''get volume of sphere with r'''
        return 4 * pi * r ** 3 / 3
```

```
    def rectangular_box(self, a, b, c):
        '''get volume of rectangular_box with r'''
        return a * b * c

    def right_circular_cone(self, r, h):
        '''get volume of right_circular_cone with r, h'''
        return (1/3) * pi * r ** 2 * h

    def cube(self, l):
        '''get volume of cube with l'''
        return l ** 3

    def cylinder(self, r, h):
        '''get volume of cylinder with r, h'''
        return pi * r ** 2 * h

    def frustum_of_a_cone(self, r, R, h):
        '''get volume of frustum of a cone with r, R, h'''
        return (1/3) * pi * h * (r**2 + r*R + R**2)

ga = geometry_area()
print("square area(4) : %d"%ga.square(4))
gb = geometry_busbar()
print("right circular cone busbar(1,1) : %f"%gb.right_circular_cone(1,2))
gp = geometry_perimeter()
print("circle perimeter(5) : %f"%gp.circle(5))
gpy = geometry_pythagorean()
print("pythagorean theorem(4,5) : %f"%gpy.pythagorean_theorem(4,5))
gv = geometry_volume()
print("cylinder volume(3,2) : %f"%gv.cylinder(3,2))
```

```
square area(4) : 16
right circular cone busbar(1,1) : 2.236068
circle perimeter(5) : 31.415927
pythagorean theorem(4,5) : 6.403124
cylinder volume(3,2) : 56.548668
```

Problem 5

[2]:
```
class Msg:
    ffrom="From : "
    to="To : "
    content="Content : "

    def __init__(self, f, t):
        self.ffrom=self.ffrom+f
        self.to=self.to+t
```

```
    def add_content(self, c):
        self.content=self.content+c

    def __str__(self):
        return "%s\n%s\n%s"%(self.ffrom, self.to, self.content)

msg=Msg('Lee','Heo')
msg.add_content("Dear friend, I would like to ....")
print(str(msg))
```

```
From : Lee
To : Heo
Content : Dear friend, I would like to …
```

Problem 7

[38]:
```
class Letter:
    def __init__(self, letterfrom, letterto):
        self.letterfrom = letterfrom
        self.letterto = letterto

    def addLine(self, line):
        for i in range(line):
            print("%d line of body\n"%(i+1))

    def get_text(self):
        print("Dear : %s\n"%self.letterto)
        self.addLine(3)
        print("Sincerely,\n")
        print("sender : %s\n"%self.letterfrom)

le = Letter('Lee','Earth')
le.get_text()
```

```
Dear : Earth

1 line of body

2 line of body

3 line of body

Sincerely,

sender : Lee
```

Problem 6

```
root@K-VirtualBox:/home/kunuk/opensource/om# ls
division.c  main.c    multiplication.c  subtraction.c
division.h  Makefile  multiplication.h  subtraction.h
root@K-VirtualBox:/home/kunuk/opensource/om# cat Makefile
CC = gcc
CFLAG = -l.

MatrixCalc: subtraction.o multiplication.o division.o main.o
        $(CC) -o MatrixCalc 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 MatrixCalc

root@K-VirtualBox:/home/kunuk/opensource/om# 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@K-VirtualBox:/home/kunuk/opensource/om# ls
division.c  main.c    MatrixCalc        multiplication.o  subtraction.o
division.h  main.o    multiplication.c  subtraction.c
division.o  Makefile  multiplication.h  subtraction.h
root@K-VirtualBox:/home/kunuk/opensource/om# ./MatrixCalc
A 행렬의 행(row) : 2
A 행렬의 열(column) : 2
>> A 행렬 입력
>> 1
2
>> 3
4

B 행렬의 행(row) : 2
B 행렬의 열(column) : 2
>> B 행렬 입력
>> 1
2
>> 3
4

1. Subtraction / 2. Multiplication / 3.Element-wise Division / 0. Quit
>> 1
result
0       0
0       0
1. Subtraction / 2. Multiplication / 3.Element-wise Division / 0. Quit
>> 2
result
7       10
15      22
1. Subtraction / 2. Multiplication / 3.Element-wise Division / 0. Quit
>> 3
result
1       1
1       1
```

Problem 8

```
root@K-VirtualBox:/home/kunuk/opensource# 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@K-VirtualBox:/home/kunuk/opensource# ./p8.sh
The size : 7
The array : 1 2 3 4 5 6 7
```

Problem 9

```
root@K-VirtualBox:/home/kunuk/opensource# cat p9.sh
for((v1 = 12; v1 < 34; v1++))
do
        echo "$v1"
done > output
root@K-VirtualBox:/home/kunuk/opensource# ./p9.sh
root@K-VirtualBox:/home/kunuk/opensource# ls
output  p8.sh  p9.sh
root@K-VirtualBox:/home/kunuk/opensource# 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@K-VirtualBox:/home/kunuk/opensource# cat p10.sh
for file in *
do
        echo $file
done
root@K-VirtualBox:/home/kunuk/opensource# ./p10.sh
output
p10.sh
p8.sh
p9.sh
root@K-VirtualBox:/home/kunuk/opensource# ls
output  p10.sh  p8.sh  p9.sh
```