ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΜΕ PYTHON –ΟΜΑΔΑ ΘΕΜΑΤΩΝ Α

ΑΣΚΗΣΗ 1 (20%)

α) Γράψτε μια συνάρτηση recEq(n) με παράμετρο έναν ακέραιο αριθμό n, η οποία υπολογίζει και επιστρέφει ως λίστα την ακολουθία $a_k = \cos((2k+3)\pi/2)a_{k-1}$ με $a_0 = 1$, για k = 0, 1, …, n.

```
def recEq(n):
    import math
    a = []
    k = 0
    while k <= n:
        if k == 0:
            a.append(1)
        else:
            a.append(math.cos((2*k+3)*math.pi/2.) * a[-1])
        k += 1
    return a
```

β) Να γραφεί πρόγραμμα στη γλώσσα προγραμματισμού python, το οποίο να λαμβάνει από το χρήστη μια φράση με χαρακτήρες του αγγλικού αλφαβήτου και να εμφανίζει στην οθόνη τη φράση διατηρώντας μόνο τους αγγλικούς κεφαλαίους χαρακτήρες.

```
s=input("Type:")
t=[]
for i in range (len(s)):
    if( (s[i]>='A' and s[i]<='Z') or s[i]<=' '):
        t.append(s[i])
print (''.join(t))
```

ΑΣΚΗΣΗ 2 (20%)

α) Δίνεται η λογική συνάρτηση F = ((A XOR B) AND (C OR D))

Γράψτε πρόγραμμα στη γλώσσα προγραμματισμού python που θα διαβάζει τις τιμές των μεταβλητών A, B, C, D και θα εμφανίζει στην οθόνη την έξοδο F.

```
a = int(input("Type value of A (0 - 1): "))
b = int(input("Type value of B (0 - 1): "))
c = int(input("Type value of C (0 - 1): "))
d = int(input("Type value of D (0 - 1): "))
print ((a^b) & (c|d))
```

β) Να γραφεί πρόγραμμα στη γλώσσα προγραμματισμού python, το οποίο να υπολογίζει και να εκτυπώνει το άθροισμα 9+98+987+…+9876543210.

```
term=9
sum=0
for i in range(1,11):
    sum+=term
    term=(term*10) + 9 - i
print (sum)
```

ΑΣΚΗΣΗ 3 (20%)

α) Να γραφεί πρόγραμμα στη γλώσσα προγραμματισμού python, το οποίο θα διαβάζει 30 αριθμούς, και θα αποθηκεύει σε αρχείο (με όνομα της επιλογής σας) τη μέση τιμή των 4 μικρότερων αριθμών.

```
a=[]
for i in range(30):
        a.append(int(raw_input("Type value: ")))
a.sort()
outfile = open('numbers.txt', 'w')
outfile.write(str((a[0]+a[1]+a[2]+a[3])/4.)+'\n')
outfile.close()
```

β) Να γραφεί συνάρτηση σε γλώσσα Python, η οποία δέχεται ως είσοδο μια λίστα από μονοψήφιους θετικούς ακεραίους αριθμούς και εκτυπώνει για κάθε εμφανιζόμενο αριθμό στη λίστα το πλήθος φορών εμφάνισής του.

```
def histogram(s):
    d = dict()
    for c in s:
    if c not in d:
      d[c] = 1
    else:
      d[c] += 1
    return d
```

ΑΣΚΗΣΗ 4 (20%)

α) Γράψτε συνάρτηση σε γλώσσα python που δέχεται ως όρισμα ένα θετικό ακέραιο αριθμό και επιστρέφει την αξία του στο οκταδικό.

```
decimal = int(input("Enter a decimal integer: "))
if decimal == 0:
   print (0)
else:
   bstring = ''
   while decimal > 0:
      remainder = decimal%8
      decimal = decimal//8
      bstring = str(remainder) + bstring
   print (bstring)
```

β) Γράψτε σε γλώσσα python πρόγραμμα το οποίο θα διαβάζει από το πληκτρολόγιο τα στοιχεία ενός πίνακα διαστάσεων 10 x 10 και αν ο πίνακας είναι αντισυμμετρικός θα εμφανίζει αντίστοιχο μήνυμα.

```
N=10
arr = [[0]*(N) for i in range(N)]
asym = 1
for i in range(N):
   for j in range(N):
      arr[i][j]=float(input("Type value:"))
for i in range(N):
   for j in range(i+1,N):
      if (arr[i][j] != -arr[j][i]):
         asym = 0
         break
if (asym == 0):
   print ("The matrix is not antisymmetric")
else:
   print("The matrix is antisymmetric ")
```

ΑΣΚΗΣΗ 5 (20%)

α) Γράψτε κλάση Fraction σε γλώσσα προγραμματισμού python η οποία θα παίρνει ως είσοδο τον αριθμητή και τον παρονομαστή ενός κλάσματος και κατά την αρχικοποίησή της θα τυπώνει το κλάσμα απλοποιημένο.

```
class Fraction:
   def gcd(self, n, d):
      while d:
         n, d = d, n%d
```

```
        return n
    def __init__(self, n, d):
        self.num = int(n / self.gcd(abs(n), abs(d)))
        self.denom = int(d / self.gcd(abs(n), abs(d)))
        if self.denom < 0:
            self.denom = abs(self.denom)
            self.num = -1*self.num
        elif self.denom == 0:
            raise ZeroDivisionError
        if (self.denom!=1):
            print (self.num,"/",self.denom)
        else:
            print (self.num)
```

β) Γράψτε μια συνάρτηση two_ranking με δυο παραμέτρους: (1) candidates, που είναι μια λίστα υποψηφίων να εκλεγούν π.χ. ['cand1','cand2','cand3','cand4'], και (2) voters, που είναι ένα λεξικό με κλειδιά κάποιους ψηφοφόρους (voters) και τιμές μια πλειάδα (tuple) αποτελούμενη από 2 υποψήφιους κατά σειρά προτίμησης π.χ. {'voter1':('cand1','cand2'), 'voter2':('cand3','cand4'), 'voter3':('cand2','cand3')}. Η συνάρτηση two_ranking, για κάθε ψηφοφόρο, θα δίνει 2 ψήφους στον υποψήφιο που ο ψηφοφόρος αυτός τον επιλέγει πρώτο και 1 ψήφο στον υποψήφιο που τον επιλέγει δεύτερο. Στο τέλος, η συνάρτηση two_ranking εκτυπώνει υποψήφιους μαζί με τους συνολικούς ψήφους που πήραν από την ψηφοφορία.

```
def two_ranking(candidates,voters):
    results={}
    for key,val in voters.items():
        u=2
        for candi in val:
            if candi not in results:
                results[candi]=u
            else:
                results[candi]+=u
            u-=1
    result={}
    for i,v in results.items():
        if v not in result:
            result[v]=[i]
        else:
            result[v].append(i)
    for i in sorted(result.keys(),reverse=True):
        for j in result[i]:
            print ('O',j, 'phre',results[j],'psifous')
```