ΑΡΧΗ 1ΗΣ ΣΕΛΙΔΑΣ

ΠΡΟΤΕΙΝΟΜΕΝΑ ΘΕΜΑΤΑ 3

ΛΥΣΕΙΣ

Α1.

1. Σωστό
2. Λάθος
3. Σωστό
4. Σωστό
5. Σωστό

Α2.

1. δ
2. γ
3. ε
4. στ
5. β

Α3.

X	Y	Z	X>Y and not(Y<=Z)	X>2 and Y<4 or Z>=5
10	5	3	True	False
2	5	6	False	True
12	3	5	False	True

Α4.

```
def binarySearch( array, key ) :
        first = 0
        last = len(array) – 1
        pos = -1
        while first <= last and pos == –1 :
                mid = ( first + last ) / 2
                if array[ mid ] == key :
                        pos = mid
                elif array[ mid ] < key :
                        first = mid + 1
                else :
                        last = mid – 1
        return pos
```

1) -1

2) ==

3) mid

4) <

ΤΕΛΟΣ 1ΗΣ ΑΠΟ 6 ΣΕΛΙΔΕΣ

ΑΡΧΗ 2ΗΣ ΣΕΛΙΔΑΣ

B1.

```
def enqueue(queue, item) :
        queue = queue.append( item )

def dequeue(queue) :
        return queue.pop( 0 )

def isEmpty(queue) :
        return len(queue) == 0

def createQueue( ) :
        return [ ]
```

B2.

```
10
20
40
```

B3.

```
class Queue :
        def __init__(self ) :
                self.items = [ ]
        def enqueue(self, item) :
                self.items.append( item )
        def dequeue(self) :
                return self.items.pop( 0 )
        def isEmpty(self) :
                return len(self.items)==0
```

B4.

```
A=Queue()
A.enqueue(10)
A.enqueue(20)
print A.dequeue()
A.enqueue(40)
if not A.isEmpty():
        A.enqueue(87)
        A.enqueue(-6)
print A.dequeue()
print A.dequeue()
```

ΑΡΧΗ 3ΗΣ ΣΕΛΙΔΑΣ

Γ.

```
# **-coding:cp1253  -*-

# -*- coding: cp1253 -*-
# Η παραπάνω εντολή είναι για να βγάζει σωστά τα ελληνικά

#(Γ1)
def vres_kostos(kibika):
        if kibika>=0 and kibika<=40:
#Αν τα κυβικά νερού είναι από 0 μέχρι και 40 θα τα πληρώσω όλα προς 0.25
                kostos_nerou=kibika*0.25
#Αν είναι πάνω από 40 μέχρι και 90 θα πληρώσω τα πρώτα 40 προς 0.25
#και όσα περισεύουν (kibika-40) επί 0.30
        elif kibika>40 and kibika<=90:
                kostos_nerou=40*0.25+(kibika-40)*0.30
#Αν είναι πάνω από 90 θα πληρώσω τα πρώτα 40 προς 0.25
#τα επόμενα 50 (από 40 μέχρι και 90 ειναι 50) προς 0.30 και
#όσα περισεύουν (kibika-90) επί 0.36 , έχω πληρώσει ήδη με 0.25 τα πρώτα 40
# και με 0.30 τα επόμενα 50, άρα έχω πληρώσει ήδη τα πρώτα 90
        elif kibika>90:
                kostos_nerou=40*0.25+50*0.30+(kibika-90)*0.36

        return kostos_nerou

#(Γ2)
EP=[]
ON=[]
KYB=[]
#(Γ4)
KOSTOSN=[]
ep=raw_input("Δώσε το Επώνυμο")
#(Γ3)
while ep!="TELOS":
#(Γ2)
        on=raw_input("Δώσε το όνομα")
        kyb=float(input("Δώσε τα κυβικά νερού"))
        EP.append(ep)
        ON.append(on)
        KYB.append(kyb)
        #(Γ4)
        kostos_nerou=vres_kostos(kyb)
        print "Ο ", ep,on," κατανάλωσε ", kyb," κυβικά νερού που κοστίζουν ",kostos_nerou," €"
```

ΤΕΛΟΣ 3ΗΣ ΑΠΟ 6 ΣΕΛΙΔΕΣ

ΑΡΧΗ 4ΗΣ ΣΕΛΙΔΑΣ

```
        KOSTOSN.append(kostos_nerou)

        ep=raw_input("Δώσε το Επώνυμο")

#(Γ5)
def bubbleSort4(A,B,C,D):
        N=len(A)
        for i in range(N-1):
                for j in range(N-1,i,-1):
                        if A[j]<A[j-1]:
                                A[j],A[j-1]=A[j-1],A[j]
                                B[j],B[j-1]=B[j-1],B[j]
                                C[j],C[j-1]=C[j-1],C[j]
                                D[j],D[j-1]=D[j-1],D[j]

bubbleSort4(KOSTOSN,EP,ON,KYB)
print "Οι 3 πελάτες με το μεγαλύτερο κόστος νερού είναι κατά σειρά:"
print EP[-1],ON[-1]," ΚΥΒΙΚΑ: ",KYB[-1], " ΚΟΣΤΟΣ ΝΕΡΟΥ: ",KOSTOSN[-1]
print EP[-2],ON[-2]," ΚΥΒΙΚΑ: ",KYB[-2], " ΚΟΣΤΟΣ ΝΕΡΟΥ: ",KOSTOSN[-2]
print EP[-3],ON[-3]," ΚΥΒΙΚΑ: ",KYB[-3], " ΚΟΣΤΟΣ ΝΕΡΟΥ: ",KOSTOSN[-3]


#(Γ6)
#Θα πρέπει να διατρέξω ξανά τις λίστες
for i in range(len(EP)):
        foros=80*KOSTOSN[i]/100.0
        teliko_poso=KOSTOSN[i]+foros
        print EP[i],ON[i]," ΦΟΡΟΣ: ", foros, "ΤΕΛΙΚΟ ΠΟΣΟ: ",teliko_poso



Δ.
# **-coding:cp1253  -*-

#(Δ3)
EP=[]
ON=[]
ALMA1=[]
ALMA2=[]
XWRA=[]

#(Δ2)
for i in range(80):
```

ΑΡΧΗ 5ΗΣ ΣΕΛΙΔΑΣ

```
        #(Δ1)
        ep=raw_input("Δώσε το επώνυμο")
        on=raw_input("Δώσε το όνομα")
        alma1=float(input("Δώσε τη βαθμολογία για το πρώτο άλμα"))
        alma2=float(input("Δώσε τη βαθμολογία για το δεύτερο άλμα"))
        xwra=raw_input("Δώσε τη χώρα του αθλητή")
        #(Δ3)
        EP.append(ep)
        ON.append(on)
        ALMA1.append(alma1)
        ALMA2.append(alma2)
        XWRA.append(xwra)


#(Δ4)
SYN=[]
# Θα πρέπει να διατρέξω τις λίστες ξανά αφού πρέπει να αθροίσω
# για κάθε αθλητή τα δύο του άλματα
for i in range(len(EP)): #θα μπορούσε και range(80)
        synolo=ALMA1[i]+ALMA2[i]
        SYN.append(synolo)
        print "Ο αθλητής ", EP[i],ON[i]," έχει συνολική βαθμολογία: ",synolo


#(Δ5)
SUMG=0.0
for i in range(len(EP)):
        if XWRA[i]=="ΓΕΡΜΑΝΙΑ":
                SUMG=SUMG+SYN[i]
print "Το άθροισμα της συνολικής βαθμολογίας της Γερμανίας είναι ",SUMG

#(Δ6)
#Στην αρχή θεωρώ οτι ο πρώτος είναι ο μέγιστος
MAX=SYN[0]
MAXEP=EP[0]
MAXON=ON[0]
#Διατρέχω ξανά τις λίστες θα πρέπει να εξε΄τασω όλα τα στοιχεία του πίνακα SYN
for i in range(len(EP)):
#Αν βρω κάποιον με μεγαλύτερη τότε αυτός είναι ο νέος MAX
# ενώ κρατάω και το επώνυμο και το όνομά του
        if SYN[i]>MAX:
                MAX=SYN[i]
                MAXEP=EP[i]
```

ΑΡΧΗ 6ΗΣ ΣΕΛΙΔΑΣ

```
                MAXON=ON[i]
print " Ο αθλητής ", MAXEP,MAXON, " έχει τη μέγιστη συνολική βαθμολογία που είναι ", MAX

#(Δ6)
m1=0
for i in range(len(EP)):
        if ALMA2[i]>ALMA1[i]:
                m1=m1+1
print "Οι αθλητές που είχαν μεγαλύτερη βαθμολογία στο δεύτερο άλμα σε σχέση με το πρώτο
είναι ",m1
```