Οικονομικό Πανεπιστήμιο Αθηνών, Τμήμα Πληροφορικής
Μάθημα: Εισαγωγή στον Προγραμματισμό Υπολογιστών
Ακαδημαϊκό έτος: 2018–19
Διδάσκων: Α. Δημάκης

Τελική Εξέταση Σεπτεμβρίου: διάρκεια 2 ώρες

Σύνολο μονάδων: 11, Άριστα: 10

ΚΑΛΗ ΕΠΙΤΥΧΙΑ!

Επίθετο:_____ Όνομα:_____ ΑΜ: _____

1η άσκηση. (2) Συμπληρώστε τα κενά. Συμπληρώστε ΜΗΝΥΜΑ ΛΑΘΟΥΣ εάν πρόκειται να εμφανιστεί κάποιο μήνυμα λάθους, ή ΚΕΝΟ αν δεν εμφανιστεί κάτι.

```
>>> x = 3
>>> y = 'hello'
>>> x*y
'hellohellohello'
>>> def x(y):
        return y + 'world'
>>> x('one')
'oneworld'
>>> x + 1
ΜΗΝΥΜΑ ΛΑΘΟΥΣ
>>> def f(x, y):
        def g(y):
            return x(y)
        return g
>>> f(lambda x: x+1, 3)(0)
1
>>> g = lambda x,y: x(y)
>>> g(lambda x: x+1, 3)
4
>>> [x*x for x in range(1, 5) if x % 2]
[1, 9]
>>> ls = [1, 2, 3]
>>> sum(x*y for x in ls for y in ls if x != y)
22
```

2η άσκηση. (0.5) Συμπληρώστε στο παρακάτω κενό τον ορισμό της συνάρτησης `is_leap_year` η οποία επιστρέφει `True` ή `False` ανάλογα εάν το όρισμα είναι δίσεκτο έτος ή όχι. (Ένα έτος είναι δίσεκτο εάν διαιρείται με το 4 αλλά όχι με το 100. Όμως, εάν διαιρείται με το 400 θεωρείται δίσεκτο.) Για παράδειγμα, η κλήση `is_leap_year(2019)` επιστρέφει `False`, η `is_leap_year(2020)` επιστρέφει `True` και η `is_leap_year(2000)` επιστρέφει `True`.

```
def is_leap_year(y):
    return (y % 4 == 0 and not y % 100 == 0) or y % 400 == 0
```

3η άσκηση. Στην άσκηση αυτή θα δώσετε τρεις εναλλακτικές υλοποιήσεις της συνάρτησης `count_leap_years` η οποία επιστρέφει το πλήθος των δίσεκτων ετών από το έτος 1 μέχρι και το έτος που δίδεται στο όρισμα της. Πχ., η κλήση `count_leap_years(3)` επιστρέφει 0 ενώ η `count_leap_years(8)` επιστρέφει 2 (αφού τα έτη 4 και 8 είναι δίσεκτα). *Σε κάθε υλοποίηση, μπορείτε να χρησιμοποιείτε κλήσεις στη συνάρτηση `is_leap_year` που περιγράφεται στη 2η άσκηση.*

a) **(1)** *Υλοποίηση με επαναληπτικούς υπολογισμούς*: Συμπληρώστε τα κενά του κώδικα που ακολουθεί χρησιμοποιώντας εντολές `while` ή `for` – χωρίς αναδρομικές κλήσεις συναρτήσεων και *comprehensions**.

* *comprehensions είναι εκφράσεις της μορφής* `[x*x for x in range(1, 10)]`, `(x+1 for x in [1,2,3] if x % 1 == 1)`, *κτλ.*

```
def count_leap_years(n):
    """ΥΛΟΠΟΙΗΣΗ ΜΕ ΕΝΤΟΛΕΣ ΕΠΑΝΑΛΗΨΗΣ while ή for """
    """ΣΥΜΠΛΗΡΩΣΤΕ ΤΟ ΣΩΜΑ ΤΗΣ ΣΥΝΑΡΤΗΣΗΣ."""
    count = 0
    for y in range(1, n+1):
        count += is_leap_year(y)
    return count
```

b) **(1)** *Υλοποίηση με comprehensions*: Συμπληρώστε τα κενά του κώδικα που ακολουθεί χρησιμοποιώντας comprehensions της Python (πχ, list comprehension) – χωρίς εντολές επανάληψης `while`, `for` και αναδρομικές κλήσεις συναρτήσεων. Η συνάρτηση `sum` της Python μπορεί να σας φανεί χρήσιμη!

```
def count_leap_years(n):
    """ΥΛΟΠΟΙΗΣΗ ΜΕ COMPREHENSIONS """
    """ΣΥΜΠΛΗΡΩΣΤΕ ΤΟ ΣΩΜΑ ΤΗΣ ΣΥΝΑΡΤΗΣΗΣ."""
    return sum(is_leap_year(y) for y in range(1, n+1))
```

c) **(1)** *Υλοποίηση με αναδρομή*: Συμπληρώστε τα κενά του κώδικα που ακολουθεί χρησιμοποιώντας αναδρομικές κλήσεις συναρτήσεων – χωρίς εντολές επανάληψης `while`, `for` και comprehensions.

```
def count_leap_years(n):
    """ΥΛΟΠΟΙΗΣΗ ΜΕ ΑΝΑΔΡΟΜΗ """
    """ΣΥΜΠΛΗΡΩΣΤΕ ΤΟ ΣΩΜΑ ΤΗΣ ΣΥΝΑΡΤΗΣΗΣ."""
    if n == 1:
        return 0
    else:
        return count_leap_years(n-1) + is_leap_year(n)
```

4η άσκηση. Εδώ θα υλοποιήσετε την αφαίρεση ενός διακόπτη (toggle) ο οποίος μπορεί να βρίσκεται σε δύο λογικές καταστάσεις (`True` ή `False`) μεταξύ των οποίων εναλλάσσεται. Συμπληρώστε τα κενά του κώδικα που ακολουθεί έτσι ώστε να παράγεται το αναγραφόμενο αποτέλεσμα.

```
>>> class Toggle:
        """(1.5) ΣΥΜΠΛΗΡΩΣΤΕ ΤΟΝ ΟΡΙΣΜΟ ΤΗΣ ΤΑΞΗΣ Toggle."""
        def __init__(self, x):
            self.state = x
        def switch(self):
            self.state = not self.state
            return self.state
```

```
>>> t1 = Toggle(False) # Στο όρισμα δίνεται η αρχική τιμή
>>> t1.switch() # αλλάζει τιμή και την επιστρέφει
True
>>> t1.switch()
False
>>> t1.switch()
True
>>> t2 = Toggle(True)
>>> t2.switch()
False
>>> t1.switch()
False
```

```
>>> # (2) ΣΥΜΠΛΗΡΩΣΤΕ ΚΩΔΙΚΑ ΣΤΟ ΚΕΝΟ:
>>> def make_toggle(x):
        t = Toggle(x)
        def switch_func():
            return t.switch()
        return switch_func
```

```
>>> t1 = make_toggle(False)
>>> t1()
True
>>> t1()
False
>>> t1()
True
>>> t2 = make_toggle(True)
>>> t2()
False
>>> t1()
False
```

5η άσκηση. (2) Συμπληρώστε τα κενά στον κώδικα της συνάρτησης `consecutive_a(s)` η οποία επιστρέφει το μέγιστο πλήθος συνεχόμενων `'a'` που περιέχονται στο αλφαριθμητικό (string τύπου `str`) `s`.

```
def consecutive_a(s):
    """Επιστρέφει το μέγιστο πλήθος συνεχόμενων 'a'.
    >>> consecutive_a('aardvark')
    2
    >>> consecutive_a('Kaaawa')
    3
    >>> consecutive_a('AAARGGGHHHH')
    0
    """
    """ΣΥΜΠΛΗΡΩΣΤΕ ΤΟ ΣΩΜΑ ΤΗΣ ΣΥΝΑΡΤΗΣΗΣ."""
    def a_prefix(i):
        """Υπολογίζει το πλήθος συνεχόμενων 'a' από τη θέση i του s
        """
        base = i
        while i < len(s) and s[i] == 'a':
            i += 1
        return i - base

    max_count, i = 0, 0
    while i < len(s):
        count = a_prefix(i)
        max_count = max(max_count, count)
        i += max(count, 1)
    return max_count
```