UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Examination in:	INF1100 — Introduction to programming with scientific applications
Day of examination:	Thursday, October 13, 2011
Examination hours:	15.00 – 19.00.
This examination set consists of 8 pages.	
Appendices:	None.
Permitted aids:	None.

Make sure that your copy of the examination set is complete before you start solving the problems.

- Read through the complete exercise set before you start solving the individual exercises. If you miss information in an exercise, you can provide your own reasonable assumptions as long as you explain that in detail.
- The maximum possible score on the exam is 25 points. The maximum number of points is listed for each exercise (a correct answer of a subquestion ((a), (b), etc.) gives 1 point).

Exercise 1 (10 points)

What will be the output of the `print` statement in the programs below?

(a)

```
q = -0.17
print '%.1f' % q
```

Solution:

```
-0.2
```

(b)

(Continued on page 2.)

```
for i in range(-2, 6, 3):
    print i-1
```

Solution:

```
-3
0
3
```

(c)

```
a = 11
b = 10/a + 1
print b
```

(Assume Python version 2.x, e.g., version 2.7.)

Solution:

```
1
```

10/11 gives integer division, resulting in 0, for Python versions 1.x and 2.x. With Python 3.x, 10/11 is 0.9090909090909091.

(d)

```
A = [-1, 1, 5] + ['plot1.eps', "plot2.png"]
del A[1]
print A
```

Solution:

```
[-1, 5, 'plot1.eps', 'plot2.png']
```

(Python prints strings with single quotes, but this is not important in the answer on the exam.)

(e)

```
A = [-1, 9, 2, 5, 19, 21, 33]
print A[3:-2]
```

Solution:

```
[5, 19]
```

(f)

(Continued on page 3.)

```
integers = []
value = 0
stop = 1
increment = 1
while value <= stop:
    integers.append(value)
    value += increment
for v in integers:
    print v
```

Solution:

```
0
1
```

(g)

```
print [0.2*(i+1) for i in range(2)]
```

Solution:

```
[0.2, 0.4]
```

(h)

```
from math import sqrt

def f(x):
    return a*sqrt(x)

x = 6; a = -2
print '%g' % f(x + a)
```

Solution:

```
-4
```

(i)

```
for i in range(2, 4):
    for j in range(i-1, i+2):
        if i != j:
            print i, j+1
```

Solution:

(Continued on page 4.)

```
2 2
2 4
3 3
3 5
```

(j)

```
def func1(x, y):
    if x > 0:
        print 'quadrant I or IV'
    if y > 0:
        print 'quadrant I or II'

def func2(x, y):
    if x > 0:
        print 'quadrant I or IV'
    elif y > 0:
        print 'quadrant II'

for x, y in (-1, 1), (1, 1):
    func1(x,y)
    func2(x,y)
```

Solution:

```
quadrant I or II
quadrant II
quadrant I or IV
quadrant I or II
quadrant I or IV
```

Exercise 2 (3 points)

The following formula finds the time t it takes (in seconds) for the center of the yolk in an egg to reach the temperature T_y (in Celsius degrees):

$$t = \frac{M^{2/3} c \rho^{1/3}}{K\pi^2(4\pi/3)^{2/3}} \ln\left[0.76\frac{T_o - T_w}{T_y - T_w}\right] .$$

The symbols M, ρ, c, and K reflect physical properties of the egg (mass, density, specific heat capacity, and thermal conductivity, respectively.). Relevant values are $M = 47$ g for a small egg, $M = 57$ g for a medium-sized egg, and $M = 67$ g for a large egg; $\rho = 1.038\ \mathrm{g\,cm^{-3}}$; $c = 3.7\ \mathrm{J\,g^{-1}\,K^{-1}}$; and $K = 5.4 \cdot 10^{-3}\ \mathrm{W\,cm^{-1}\,K^{-1}}$. Furthermore, T_w is the temperature (in C degrees) of the boiling water ($T_w = 100$ C), and T_o is the original temperature (in C degrees) of the egg before it is put in the boiling water.

(Continued on page 5.)

Implement the formula for t in a Python function `egg(M, To=20, Ty=70)`. Insert a doc string in the function for explaining the purpose of the calculation, what the arguments are, and what is returned. You do not need to call the `egg` function in this exercise.
Solution:

```
from math import pi, log as ln

def egg(M, To=20, Ty=70):
    """
    Return the cooking time of an egg with mass
    M at temperature To. The target yolk temperature
    is Ty.
    """
    c = 3.7       # specific heat capacity
    rho = 1.038   # density
    K = 5.4E-03   # thermal conductivity
    Tw = 100      # boiling water temperature

    t = (M**(2.0/3)*c*rho**(1./3))/\
        (K*pi**2*(4*pi/3)**(2./3))*ln(0.76*\
        (To - Tw)/(Ty - Tw))
    return t
```

Exercise 3 (3 points)

Write a program that prints out a nicely formatted table of t values, computed by the `egg` function in Exercise 2, for all combinations of the following parameters:

- M for small, medium-sized, and large egg
- T_o as 4, 12, 20, and 30 degrees Celsius
- T_y as 63 and 70 degrees Celsius (corresponding to a soft and hard boiled egg, respectively)

Exclude the combination of a soft boiled egg and $T_o = 25$ C or $T_o = 30$ C. Hint: Use nested `for`-loops with an `if`-test.
Solution:

```
for M in [47, 57, 67]:
    for To in [4, 12, 20, 30]:
        for Ty in [63, 70]:
            if not (Ty == 63 and (To == 25 or To == 30)):
                t = egg(M, To, Ty)
```

(Continued on page 6.)

```
print 'M=%2.0f g, To=%2.0f C, '\
      'Ty=%2.0f C, cooking time: '\
      '%.1f min' % (M, To, Ty, t/60)
```

Exercise 4 (4 points)

The sine function can be approximated by a polynomial according to the following formula:

$$\sin x \approx S(x;n) = \sum_{j=0}^{n} (-1)^j \frac{x^{2j+1}}{(2j+1)!}\,. \tag{1}$$

The expression $(2j+1)!$ is the factorial and can be computed by the `factorial` function in the `math` module.

Write a Python function `S(x, n)` that computes and returns the value of $S(x;n)$. Show how you can verify the implementation by comparing the result of `S(x, n)` with the known expressions x and $x - \frac{1}{6}x^3$ for $n = 0$ and $n = 1$.

Solution:

```
from math import factorial

def S(x, n):
    s = 0
    for j in range(n+1):
        s = s + (-1)**j*x**(2*j+1)/factorial(2*j+1)
    return s

def verify():
    x = 1.2  # some arbitrary test value
    S0 = x
    S1 = x - x**3/6.0
    tol = 1E-14
    correct = True
    # Never test S(x,0) == S0, always use
    # abs of difference and a tolerance
    if abs(S(x, 0) - S0) > tol:
        correct = False
    if abs(S(x, 1) - S1) > tol:
        correct = False
    return correct

print 'The program is correct:', verify()
```

(Continued on page 7.)

Exercise 5 (2 points)

Plot $\sin x$ on $[0, 4\pi]$ together with the approximations $S(x;1)$, $S(x;2)$, $S(x;3)$, $S(x;6)$, and $S(x;12)$, where $S(x;n)$ is defined and implemented in Exercise 4. Make sure each curve is identified with a proper label. Let the y axis cover the interval $[-2, 2]$.

Solution:

```
from scitools.std import *
x_min = 0
x_max = 4*pi
x = linspace(x_min, x_max, 100)

plot(x, sin(x), 'k-',
     x, S(x, 1), 'r-',
     x, S(x, 2), 'b-',
     x, S(x, 3), 'y-',
     x, S(x, 6), 'g-',
     x, S(x, 12), 'c-',
     axis=[x_min, x_max, -2, 2],
     title='Taylor series approximation to the sine function',
     legend=('sin(x)', 'n=1', 'n=2', 'n=3', 'n=6', 'n=12'))
```

Exercise 6 (3 points)

Make a function for solving the system of difference equations

$$p_j = p_{j-1} + q_{j-1}, \tag{2}$$
$$q_j = -x^2 \left((2j+1)2j\right)^{-1} q_{j-1}, \tag{3}$$

with initial conditions $p_0 = 0$ and $q_0 = x$. In the function, store only the newest two p_j and q_j values (i.e., do not store all the p_j and q_j values in arrays). The function should take two arguments, x and n, and return the two values p_n and $|q_n|$. Write a main program that prints out the value of p_{20} for $x = \pi$.

Solution:

```
def diffeq(x, n):
    pj_prev = 0
    qj_prev = x
    index = range(N+1)
    for j in index[1:]:
        pj = pj_prev + qj_prev
        qj = -x**2/float((2*j+1)*(2*j))*qj_prev
        pj_prev = pj
        qj_prev = qj
```

(Continued on page 8.)

```
    return pj, abs(qj)

n = 20
from math import pi
x = pi
s_n, a_n = diffeq(x, n)
print s_n
```

END