MATEMATISKA INSTITUTIONEN
STOCKHOLMS UNIVERSITET
Avd. Beräkningsmatematik
Kursledare: Lars Arvestad
Examinator: Lars Arvestad

Tentamen i
DA2005 Programmeringsteknik *och*
DA2004 Programmeringsteknik för matematiker
7.5 hp
2024-08-22

---

**Note**: An English version of this exam starts on page 5!

---

- Tentan har flervalsfrågor där minst ett svarsalternativ är korrekt. Om man svarar fel eller inte har exakt antal rätta alternativ får man noll poäng på frågan.
- Man måste bli godkänd på del A (4 rätt på 8 frågor) för att del B ska rättas.
- Del B består av frågor med varierande poäng (totalt 12 poäng).
- Inga `import` (Pythons standardbibliotek eller externa bibliotek) får användas om de inte nämns eller finns med i uppgiften. Man får använda inbyggda funktioner som `len`, `range` och `map`.
- All kod avser **Python 3**, dvs *inte* t.ex. Python 2.7
- **Hjälpmedel:** Ett A4 med så mycket information du vill. Du får skriva på båda sidorna.
- **Betygsgränser:** E: 10, D: 12, C: 14, B: 16, A: 18, av maximala 20.

---

## Del A: flervalsfrågor

*Var snäll samla svaren på del A på ett svarspapper.*

**1**. Vad skrivs ut av koden till höger?

A. 1
B. 9
C. 11
D. 19
E. Det blir ett särfall.

```
y = 10
for x in [2, -4, 3]:
    y -= x
print(y)
```

**2**. Vilken lista returneras av funktionen `fcn`?

A. `[]`
B. `[1, 2, 3, 4, 5]`
C. `[2, 4, 6, 8, 10]`
D. `[1, 4, 9, 16, 25]`
E. `[0, 1, 4, 9, 16]`

```
def fcn():
    res = []
    for i in range(5):
        res.append(i**2)
    return res
```

**3**. Vilka av dessa alternativ har en korrekt Python-syntax för att skapa en lista i variabeln `x`?

A. `x = []`
B. `x = [2 , , , 7]`
C. `x = [5, 9]`
D. `x = [[1, 2], [3, 4]]`
E. `x = [[1 ... 5],[1 ... 5]]`

**4**. Vad blir värdet i variabeln `text` när koden till höger har exekverats?

A. `'pot'`
B. `'tp'`
C. `'top'`
D. `'t'`
E. `'o'`

```
text = ''
msg= 'top'
i = 0
while i < len(msg):
    if i % 2 == 0:
        text += msg[i]
    i += 1
```

**5**. Vad innebär begreppet *typkonvertering* (även känt som *type casting*).

A. Man begränsar variabler till endast en typ. Till exempel kan man bestämma att `x` bara kan lagra heltalsvärden.
B. Man tilldelar en variabel ett värde av en ny typ. Till exempel kan man först göra `x = 1` och sedan `x = 7.3`.
C. Man tolkar om ett värde till en annan typ. Till exempel kan ett heltal göras om till ett flyttal.
D. Man gör en avrundning.
E. Man använder en if-sats till att avgöra om ett värde är giltigt eller inte.

**6**. Vad skrivs ut av koden till höger?

A. 0
B. 1
C. 4
D. 8
E. Det blir ett särfall.

```
x = 1
y = 8
while y > 0:
    y //= 2
    x *= y
print(x)
```

**7**. Vad skrivs ut av koden till höger?

A. `0`
B. `2`
C. `4`
D. `8`
E. Ett felmeddelande

```
def f(x, y):
    return x + y

def g(y):
    return f(y, y) * 2

print(g(2))
```

**8**. Vad kommer variabeln `d` ha för värde efter att koden till höger är exekverad?

A. `{'D': 1, 'A': 1, '2': 1, '0': 2, '5': 1}`
B. `{'D': 0, 'A': 1, '2': 2, '0': 3, '5': 9}`
C. `{'D': 0, 'A': 1, '2': 2, '0': 3, '5': 5}`
D. `{'D': 1, 'A': 1, '2': 1, '0': 1, '5': 4}`
E. `{0: 'D', 1: 'A', 2: '2', 3: '0', 4: '0', 5: '5'}`

```
s = 'DA2005'
d = {}
for x in s:
  d[x] = s.count(x)
```

## Del B: kodfrågor

*Var snäll använd ett papper till varje fråga i del B. Delfrågor, som 9A och 9B, får gärna lösas på samma papper.*

**9**.

A. Skriv en funktion `my_special_prod` som beräknar produkten av alla *jämna* heltal i en lista. Till exempel ska `[1,2,3,4]` resultera i $2*4 = 8$. Om det bara finns ett jämnt heltal ska det talet returneras. Om det inte finns något jämnt heltal i lista ska 0 returneras. (2p)

**Exampelanvändning:**

```
[In: ] print(my_special_prod([6,2,4]))
[Out:] 48
[In: ] print(my_special_prod([1,3,4]))
[Out:] 4
[In: ] print(my_special_prod([1,1,1]))
[Out:] 0
```

B. Anpassa `my_special_prod` från fråga A. så att den kan användas på listor som innehåller element som inte är av typen **`int`** (till exempel **`str`**, **`float`**, **`bool`**). Funktionen ska skriva ut "`Invalid datatype at index: X`" om elementet på position X i listan inte är ett heltal. (1p)

**Tips:** Du kan använda den inbyggda funktionen **`type`**`(x)` som returnerar typen på variabeln `x`.

**Exempelanvändning:**

```
[In: ] print(my_special_prod([7.0,6,2,4]))
[Out:] invalid datatype at list at index: 0
[Out:] 48
[In: ] print(my_special_prod([1,'1',1,1, False]))
[Out:] invalid datatype at list at index: 1
[Out:] invalid datatype at list at index: 4
[Out:] 0
```

**10**. Fibonacci-talen $f_0, f_1, f_2, \ldots$ är en följd av positiva heltal som börjar med $f_0 = 0$ samt $f_1 = 1$ och fortsätter med $f_i = f_{i-1} + f_{i-2}$, $i > 1$, där $f_i$ är det $i$:te Fibonacci-talet. Till exempel är $f_2 = f_1 + f_0 = 1$ and $f_3 = f_2 + f_1 = 2$. Skriv en funktion `fibonacci(n)` som returnerar de första $n$ Fibonacci-talen i en lista. (2p)

**Exempelanvändning:**

```
[In: ] fibonacci(1)
[Out:] [0]
[In: ] fibonacci(2)
[Out:] [0, 1]
[In: ] fibonacci(10)
[Out:] [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

**11**. Skriv funktionen `remove_duplicates(lst)` som tar en lista `lst` och returnerar en ny lista med upprepade element borttagna, med ordningen av de kvarvarande elementen bevarad. (2p)

**Exempelanvändning:**

```
[In: ] remove_duplicates([1, 2, 1, 2, 3, 4, 4, 5])
[Out:] [1, 2, 3, 4, 5]
```

**12**. Skriv funktionen `common_kv_pairs(d1, d2)` som tar två uppslagstabeller (dict) och returnerar en lista med nycklar för de nyckel-värde-par som är identiska i d1 och d2. (2p)

**Exempelanvändning:**

```
[In: ] d1 = {1 : 1, 'a': 2, 'c': 3}
[In: ] d2 = {1 : 2, 'a': 5, 'c': 3}
[In: ] common_kv_pairs(d1, d2)
```

```
[Out:] ['c']
[In: ] d1 = {1 : 1, 'b': 2, 'z': 3}
[In: ] d2 = {'z' : 2, 2: 5, 3: 'z'}
[In: ] common_kv_pairs(d1, d2)
[Out:] []
```

**13**. Skriv funktionen `matrix_transpose(m)` som tar en matris $m$ representerad av en lista med listor och returnerar *transponatet* (se nedan) av matrisen, dvs matrisen med rader flippade till kolumner. Du kan utgå ifrån att $m$ är korrekt (dvs listorna har samma längd). (3p)

**Exempelanvändning:**

```
[In: ] matrix = [[3, 5, 7], [1, 2, 3]]
[In: ] matrix_transpose(matrix)
[Out:] [[3, 1], [5, 2], [7, 3]]
```

**Förklaring av representationen:** Vi låter matriser representeras av en lista med rader, och varje rad ges av en lista med element. I exemplet ovan representerar `[[3, 5, 7], [1, 2, 3]]` matrisen $\begin{pmatrix} 3 & 5 & 7 \\ 1 & 2 & 3 \end{pmatrix}$ och `[[3, 1], [5, 2], [7, 3]]` representerar $\begin{pmatrix} 3 & 1 \\ 5 & 2 \\ 7 & 3 \end{pmatrix}$.

**Förklaring av transponat:** Transponatet $A^T$ av en matris $A$ är en operator som, intuitivt, "flippar" en matris över dess diagonal. Det innebär att rader blir kolumner och tvärtom. För en $2 \times 2$-matris $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ blir transponatet $A^T = \begin{pmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{pmatrix}$.

Om man till exempel transponerar $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ får man resultatet $\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$.

Transponatet är också definierat när antalet rader och kolumner skiljer sig åt. Om man till exempel transponerar $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ får man $\begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$.

# The exam in English

## Part A: Multiple choice

*Please collect your answers to part A on a single piece of paper.*

**1**. What is printed by the code to the right?

A. 1
B. 9
C. 11
D. 19
E. An exception is raised.

```
y = 10
for x in [2, -4, 3]:
    y -= x
print(y)
```

**2**. Which list does the function `fcn` return?

A. `[]`
B. `[1, 2, 3, 4, 5]`
C. `[2, 4, 6, 8, 10]`
D. `[1, 4, 9, 16, 25]`
E. `[0, 1, 4, 9, 16]`

```
def fcn():
    res = []
    for i in range(5):
        res.append(i**2)
    return res
```

**3**. Which of these options are correct Python syntax for creating a list in the variable `x`?

A. `x = []`
B. `x = [2 , , , 7]`
C. `x = [5, 9]`
D. `x = [[1, 2], [3, 4]]`
E. `x = [[1 ... 5],[1 ... 5]]`

**4**. What will be the value of the variable `text` after running the code to the right?

A. `'pot'`
B. `'tp'`
C. `'top'`
D. `'t'`
E. `'o'`

```
text = ""
msg= "top"
i = 0
while i < len(msg):
    if i % 2 == 0:
        text += msg[i]
    i += 1
```

**5**. What does the term *type casting* (also known as *type conversion*) mean?

A. You restrict variables to only one type. For example, you can decide that `x` can only store integer values.
B. You assign a variable a value of a new type. For example, you can first do `x = 1` and then `x = 7.3`.
C. You reinterpret a value to another type. For example, an integer can be converted to a floating-point number.
D. You perform a rounding operation.
E. You use an if-statement to determine whether a value is valid or not.

**6**. What is printed by the code to the right?

A. 0

B. 1

C. 4

D. 8

E. An exception is raised.

```
x = 1
y = 8
while y > 0:
    y //= 2
    x *= y
print(x)
```

**7**. What is the result of the code on the right?

A. `0`

B. `2`

C. `4`

D. `8`

E. An error message

```
def f(x, y):
    return x + y

def g(y):
    return f(y, y) * 2

print(g(2))
```

**8**. What will be the contents of `d` after the code to the right is run?

A. `{'D': 1, 'A': 1, '2': 1, '0': 2, '5': 2}`

B. `{'D': 0, 'A': 1, '2': 2, '0': 3, '5': 9}`

C. `{'D': 0, 'A': 1, '2': 2, '0': 3, '5': 5}`

D. `{'D': 1, 'A': 1, '2': 1, '0': 1, '5': 4}`

E. `{0: 'D', 1: 'A', 2: '2', 3: '0', 4: '0', 5: '5'}`

```
s = 'DA2005
d = {}
for x in s:
  d[x] = s.count(x)
```

## Part B: Coding

*Please use a separate piece of paper (or several) for each question in part B. Multipart questions such as 9A and 9B can be written on the same piece of paper.*

**9**.

A. Write a function `my_special_prod` that computes the product of all *even* integers in a list. For example, `[1,2,3,4]` results in $2 * 4 = 8$. If there is only one even integer it should return the integer itself. If there is no even integer it should return 0. (2p)

**Example use:**

```
[In: ] print(my_special_prod([6,2,4]))
[Out:] 48
[In: ] print(my_special_prod([1,3,4]))
[Out:] 4
[In: ] print(my_special_prod([1,1,1]))
[Out:] 0
```

B. Make `my_special_prod` able to handle lists that contain elements which are not of the type **`int`** (for example **`str`**, **`float`**, **`bool`**) by printing `'invalid datatype at list at index: X'` for the items where this happens, where `X` is the position (i.e., index) of the variable in the list. (1p)

**Tip:** You can use the built-in function **`type`** that returns the type of the variable.

**Example usage:**

```
[In: ] print(my_special_prod([7.0,6,2,4]))
[Out:] invalid datatype at list at index: 0
[Out:] 48
[In: ] print(my_special_prod([1,'1',1,1, False]))
[Out:] invalid datatype at list at index: 1
[Out:] invalid datatype at list at index: 4
[Out:] 0
```

**10**. The Fibonnaci sequence $f_0, f_1, f_2, \ldots$ is a sequence of positive integers that starts with the two integers $f_0 = 0$ and $f_1 = 1$ where the $i$:th integer $f_i = f_{i-1} + f_{i-2}$, $i > 1$. For example, $f_2 = f_1 + f_0 = 1$ and $f_3 = f_2 + f_1 = 2$. Write a function `fibonacci(n)` that returns the first `n` Fibonacci numbers in a list. (2p)

**Example usage:**

```
[In: ] print(fibonacci(1))
[Out:] [0]
[In: ] print(fibonacci(2))
[Out:] [0, 1]
[In: ] print(fibonacci(10))
[Out:] [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

**11**. Write a function `remove_duplicates(lst)` that takes a list `lst` and returns a new list with duplicates removed, preserving the original order of retained elements. (2p)

**Example usage:**

```
[In: ] remove_duplicates([1, 2, 1, 2, 3, 4, 4, 5])
[Out:] [1, 2, 3, 4, 5]
```

**12**. Write a function `common_kv_pairs(d1, d2)` that takes two dictionaries and returns a list of keys for which a key-value pair is identical between the two dictionaries. (2p)

**Example usage:**

```
[In: ] d1 = {1 : 1, 'a': 2, 'c': 3}
[In: ] d2 = {1 : 2, 'a': 5, 'c': 3}
[In: ] print(common_kv_pairs(d1, d2))
[Out:] ['c']
[In: ] d1 = {1 : 1, 'b': 2, 'z': 3}
[In: ] d2 = {'z' : 2, 2: 5, 3: 'z'}
[In: ] print(common_kv_pairs(d1, d2))
[Out:] []
```

**13**. Write a function `matrix_transpose(m)` that takes a matrix $m$ represented as a list of lists and returns the *transpose* (see below) of the matrix. You do not have to check that the input matrix $m$ is correct (i.e., same length on all sublists). (3p)

**Example usage:**

```
[In: ] matrix = [[3, 5, 7], [1, 2, 3]]
[In: ] matrix_transpose(matrix)
[Out:] [[3, 1], [5, 2], [7, 3]]
```

**Explanation of the representation:** We let matrices be represented as a list of rows, and each row is a list of elements. In the example above, `[[3, 5, 7], [1, 2, 3]]` represents the matrix $\begin{pmatrix} 3 & 5 & 7 \\ 1 & 2 & 3 \end{pmatrix}$ and `[[3, 1], [5, 2], [7, 3]]` represents $\begin{pmatrix} 3 & 1 \\ 5 & 2 \\ 7 & 3 \end{pmatrix}$.

**Explanation of matrix transpose:** The transpose $A^T$ of a matrix $A$ is an operator which, intuitively, 'flips a matrix over its diagonal'; that is, it switches the row and column indices of the matrix A by producing another matrix. For a two-by-two matrix $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ we have the transpose $A^T = \begin{pmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{pmatrix}$.

For example, the transpose of $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ is $\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$.

The operation is also defined when the number of rows and columns are different. For example, the transpose of $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ is $\begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$.