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

Tentamensskrivning i
DA2004 Programmeringsteknik för matematiker
7.5 hp
2020-01-17

- **Skriv tydligt:** svårlästa svar riskerar 0 poäng.
- Skriv bara på en sida av varje papper!
- Del A 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 (5 rätt på 10 frågor) för att del B ska rättas.
- Inga externa bibliotek får användas om de inte nämns i uppgiften, man får dock använda inbyggda funktioner som `len` och `range`.
- **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

*Vänligen samla svaren på del A på **ett** svarspapper.*

1. Vad är värdet på `zs` efter att vi kört koden till höger?

```
xs = [1, 5, 4]
ys = [0, 5, 6]
zs = []

for i in range(3):
    x = xs[i]
    y = ys[i]
    if x >= y:
        zs.append(x)
    else:
        zs.append(y)
```

A. `[0, 5, 4]`
B. `[1, 5, 4]`
C. `[0, 5, 6]`
D. `[1, 5, 6]`
E. Inget av alternativen A.-D. då ett särfall lyfts.

2. Givet `xs = [1,2,3,4,5,6]`, vilket eller vilka av följande `print`-uttryck skriver ut listan `[1,3,5]`?

A. `print(xs[1::2])`
B. `print(xs[0:5:2])`
C. `print([ x for x in xs if x % 2 != 0])`
D. `print([xs[0],xs[2],xs[4]])`
E. `print(xs[0] + xs[2] + xs[4])`

3. Vad skrivs ut om vi kör koden nedan?

```
d = { 'hej' : 'du', [1, 2, 3] : 17 }

print(d[[1, 2, 3]])
```

A. `du`
B. `hej`
C. `17`
D. `d`
E. Inget av alternativen A.-D. då ett särfall lyfts.

4. Vilka av följande är reserverade ord i Python 3?

A. `elif`
B. `do`
C. `raise`
D. `break`
E. `for`

5. Vad skrivs ut om vi kör koden nedan?

```
xs = [1, 2, 3]
ys = [4, 5]

print(len([ x + y for x in xs for y in ys ]))
```

A. 0
B. 5
C. 6
D. 15
E. Inget av alternativen A.-D. då ett särfall lyfts.

6. Vilken eller vilka boolska variabeltilldelningar gör att uttrycket

```
not x and (x or not y)
```

evaluerar till `True`?

A. `x = False, y = False`
B. `x = False, y = True`
C. `x = True, y = False`
D. `x = True, y = True`
E. Ingen, det evaluerar alltid till `False`.

7. Vad skrivs ut om vi kör koden till höger?

A. `-20`
B. `-10`
C. `10`
D. `30`
E. `50`

```
x = 30
y = 20

def f(x):
    if x >= y:
        return x + y
    else:
        return x - y

print(f(10))
```

8. Vad skrivs ut av följande kod?

```
foo = lambda f, x: f(x) * f(x)

print(foo(lambda x: x + x, 5))
```

A. `10`
B. `25`
C. `50`
D. `100`
E. Inget av alternativen A.-D. då ett särfall lyfts.

9. Vad händer om man kör följande kod och användaren skriver in `5`?

```
x = input("Skriv in ett tal: ")
xs = list(range(int(x)))
s = 0

while xs:
    s += xs.pop()

print(s)
```

A. `5` skrivs ut
B. `10` skrivs ut
C. `15` skrivs ut
D. Programmet fastnar i en oändlig loop
E. Inget av alternativen A.-D. då ett särfall lyfts.

10. Vad blir värdet på `xs` efter att koden till höger har körts?

A. `['foo','hej','du]`
B. `'foojehud'`
C. `'fooduhej'`
D. `'hejdufoo'`
E. `'foohejdu'`

```
xs = 'foo'

for x in ['hej','du']:
    for y in x:
        xs += y
```

Del B: kodfrågor

Vänligen använd ett papper till varje fråga i del B.

11. Inom talteori säger man att två tal är *relativt prima* om deras största gemensamma delare är `1`. Exempelvis är `12` och `25` relativt prima då de ej har några gemensamma primtalsdelare. I föreläsningarna har vi skrivit funktionen `gcd(m,n)` som returnerar största gemensamma delare av `m` och `n`. Använd denna funktion för att skriva en funktion `relPrime(m,n)` som returnerar en `bool` som säger om `m` och `n` är relativt prima eller inte.

Exempelanvändning:

```
>>> print(relPrime(12,25))
True
>>> print(relPrime(12,16))
False
```

Obs: ni behöver ej skriva `gcd` funktionen utan det räcker med `relPrime`.

12. Skriv funktionen `relPrimes(n)` som skapar en lista på alla par av relativt prima tal mellan `2` och `n`.

Exempelanvändning:

```
>>> print(relPrimes(6))
[(2, 3), (3, 4), (2, 5), (3, 5), (4, 5), (5, 6)]
```

Obs: ni får använda funktionen `relPrime` från uppgift 11 även om ni inte löst den.

13. Skriv en funktion `printLongLines(f,n)` som tar in ett filnamn `f` och ett heltal `n` och skriver ut de rader som är längre än eller lika med `n`.

Om vi har en textfil som heter `fil.txt` med innehåll som i rutan till höger ska funktionen fungera på följande sätt:

```
>>> printLongLines('fil.txt',5)
lycka
tentan!
```

```
lycka
till
på
tentan!
```

14. Skriv funktionen `addSpaces(s,n)` som tar in en sträng `s` och ett heltal `n` och sätter in `n` mellanslag mellan varje bokstav i `s`.

Exempelanvändning:

```
>>> print(addSpaces('hej',6))
h      e      j
>>> print(addSpaces('hej',2))
h  e  j
>>> len(addSpaces('hej',2))
7
>>> print(addSpaces('hej',0))
hej
```

Obs: ni får anta att strängen `s` ej är tom och endast består av ett ord (så inga blanksteg). Funktionen ska inte sätta in några mellanslag i början eller slutet av strängen.

15. Givet ett tal `n` kan vi beräkna summan av dess delare och om denna summa är mindre än `n` är talet *deficient* (sv. *bristfälligt*), om summan är lika med `n` är talet *perfect* (sv. *perfekt*) och om summan är större än `n` är talet *abundant* (sv. *överflödigt*). Skriv en funktion `numberType(n)` som tar in ett heltal `n` och skriver ut om det är deficient, perfect eller abundant.

Exempelanvändning:

```
>>> numberType(6)   # delare: [1, 2, 3]
perfect
>>> numberType(4)   # delare: [1, 2]
deficient
>>> numberType(12)  # delare: [1, 2, 3, 4, 6]
abundant
```

Obs: ni får använda funktionen `divisors(n)` som returnerar en lista med alla delare till `n` från förra tentan. Ni får även använda Python funktionen `sum` för att beräkna summan av elementen i en lista.

16. Skriv en klass `Poly` för polynom som representeras på samma sätt som i labbarna, dvs som listor av koefficienter utan nollor i slutet. Denna klass ska ha följande metoder:

- En konstruktormetod `__init__` som tar in en lista med tal och använder `drop_zeroes` från labben för att ta bort nollor i slutet.
- En metod `__str__` som använder `poly_to_string` från labben för att returnera en sträng med polynomet.

Exempelanvändning:

```
>>> p0 = Poly([2, 0, 1, 0])
>>> q0 = Poly([0, 0, 0])
>>> print(p0)
2 + x^2
>>> print(q0)
0
```

Obs: ni behöver *inte* skriva `drop_zeroes` eller `poly_to_string` på tentan utan kan bara använda funktionerna från labben.

17. Lägg till en metod `divideCoeffs` till `Poly` för att dela alla koefficienter med ett tal `n`. Om `n` är noll ska särfallet `ZeroDivisionError` lyftas.

Exempelanvändning med `p0` som ovan:

```
>>> p0.divideCoeffs(2)
>>> print(p0)
1.0 + 0.5x^2
>>> p0.divideCoeffs(0)
Traceback (most recent call last):
...
ZeroDivisionError
```

Obs: särfallet ska lyftas av `divideCoeffs` och ej genom anrop till någon annan funktion.

18. Skriv en högre ordnings funktion `compose(f,g)` som tar in två funktioner och returnerar deras sammanslagning: $(f \circ g)(x) = f(g(x))$

Exempelanvändning:

```
>>> print(compose(lambda x: x + 1, lambda x: 2 * x)(3))
7
```

Tips: returnera ett `lambda`-uttryck.

19. Skriv en *rekursiv* funktion `unzip(xs)` som tar in en lista av par `xs` och returnerar ett par av listor som innehåller de första och andra elementen i paren.

Exempelanvändning:

```
>>> print(unzip([(1,2), (3,4), (5,6)]
([1, 3, 5], [2, 4, 6])
>>> print(unzip([]))
([], [])
```

Obs: er lösning måste vara rekursiv och varken `for`, `while` eller listomfattningar får användas.

20. Sista uppgiften på den här tentan är att skriva en funktion `rövarspråk(s)` som översätter en sträng `s` till *rövarspråket*.[1] Denna översättning går till så att man efter varje konsonant lägger in ett `'o'` och sedan samma konsonant igen, till exempel byts `'b'` ut mot `'bob'` och `'k'` mot `'kok'`. Vokaler och andra tecken (mellanslag, utropstecken, etc.) är oförändrade.

Exempelanvändning:

```
>>> print(rövarspråk('anders'))
anondoderorsos
>>> print(rövarspråk('lycka till på tentan!'))
lolycockoka totilollol popå totenontotanon!
```

Obs: ni kan anta att det finns en lista `consonants` med alla konsonanter redan definierad och att strängen `s` endast innehåller gemener (dvs ”små” bokstäver).

[1] Detta kodspråk går att läsa om i Astrid Lindgrens romaner om Kalle Blomkvist.