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

Tentamensskrivning i
DA2004 Programmeringsteknik för matematiker
7.5 hp
2021-01-11

- 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 utav 8 frågor med varierande poäng (totalt 12p).
- Var noga med att **namnge** *funktioner och klasser* rätt (som det står i uppgiften) i Del B.
- Inga `import` (Pythons standardbibliotek eller externa bibliotek) får användas om de inte nämns eller finns med i uppgiften. Man får dock använda inbyggda funktioner som `len`, `range` och `map`.
- **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

1. Vilket av följande är *inte* en typ i Python?

A. **`int`**
B. **`bool`**
C. **`pair`**
D. **`str`**
E. **`dict`**

2. Vilka av följande påståenden är korrekta? (se kod till höger)

A. `d` är av datatyp uppslagstabell.
B. Vi kan använda listor som nycklar i uppslagstabeller.
C. Vi kan använda tupler som nycklar i uppslagstabeller.
D. `d` har `'hej'` och `(1, 2)` som nycklar.
E. Ett särfall lyfts om vi kör koden.

```
d = { 'hej' : [99, 101], (1, 2) : 'a' }
print(len(d))
```

3. Vad är resultatet av satsen `print([x + y for x in range(2)for y in range(2)])`?

A. `[4]`
B. `[18]`
C. `[0, 1, 2]`
D. `[0, 1, 1, 2]`
E. `[0, 1, 2, 1, 2, 3, 2, 3, 4]`
F. Ett särfall

4. Vilka slutsatser kan vi dra baserat på koden till höger?

A. `A` är en klass.
B. `A` är en subklass till `B`.
C. `t` är ett attribut till klassen `A`.
D. `p` är ett klassattribut till klassen `B`.
E. Konstruktorn ger ett särfall då parametern `r` måste ha samma identifierare (variablenamn) som attributet `t`.

```
class A(B):
    def __init__(self, r):
        self.t = r
        self.t += B.p
```

5. Vad skrivs ut av koden till höger?

A. Särfall lyfts då det inte är tillåtet att blanda `for` och `while`

B. `[(0, 'a'), (0, 'b')]`

C. `[(1, 'a'), (1, 'b')]`

D. `[(1, 'a'), (1, 'b'), (0, 'a'), (0, 'b')]`

E. `[(2, 'a'), (2, 'b'), (1, 'a'), (1, 'b')]`

```
i = 1
l = []
while i > 0:
    for y in ['a', 'b']:
        l.append((i, y))
    i -= 1
print(l)
```

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

A. `(1,2)`

B. `1`

C. `4`

D. `[1, 3]`

E. Inget av alternativen A.-D. då ett särfall lyfts.

```
res = map(lambda t: t[0], [(1,2),(3,4)])
print(list(res))
```

7. Vilka alternativ är korrekta givet koden till höger?

A. Koden skriver ut `True True`

B. Koden skriver ut `False True`

C. Koden skriver ut `False False`

D. `f1` returnerar alltid `False`

E. `f2` returnerar alltid `False`

```
def f1(x, y):
    y = False
    return x and y

def f2(x, y = False):
    return x and y

x, y = True, True
print(f1(x, y), f2(x, y))
```

8. Vad skrivs ut av koden till höger?

A. `1 5`

B. `1`

C. `3 7`

D. `1 7`

E. `[1, 7]`

F. Inget av alternativen A.-E. då ett särfall lyfts.

```
def f(x):
    i = min(min(x), len(x)-1)
    return x[i]

a = [1, 3, 4, 9]
b = [5, 10, 7]

print(f(a), f(b))
```

Del B: kodfrågor

9. **(1p)** Skriv en ekvivalent version av funktionen `flip` som använder **`for`** istället för **`while`**.

```
def flip(s):
    i = 0
    s_list = []
    while i < len(s):
        s_list.append(s[-(i+1)])
        i = i + 1
    return s_list
```

10. **(1p)** Modifiera funktionen `my_div_sum` nedan så att den i listan `data` separat hanterar heltal/flyttal, strängar, och andra typer. Flyttal/heltal som förekommer i `data` ska läggas till summan efter att ha delats med `x`. Om en sträng förekommer i `data` ska funktionen skriva ut `Fel i indata` och returnera `None`. Särfall ska inte lyftas för andra typer och ska heller inte resultera i att något läggs till i summan.

```
def my_div_sum(data, x):
    sum = 0
    for i in data:
        sum += i / x
    return sum
```

Exempel på input och resulterande output:

```
[In] : my_div_sum([8], 4)
[Out]: 2.0
[In] : my_div_sum([], 42)
[Out]: 0
[In] : my_div_sum([1, (1, 2)], 2)
[Out]: 0.5
[In] : my_div_sum([1, (1, 2), 'hello'], 2)
[Out]: Fel i indata
```

11. **(2p)** Betrakta funktionen nedan. Vad gör funktionen och vad händer om `z1` och `z2` har olika längd? Skriv en ekvivalent version av funktionen som ej använder **`enumerate`**, **`map`** och **`lambda`**.

```
def f(z1,z2):
    return [ i for i,eq in enumerate(map(lambda x,y: x==y, z1,z2)) if eq]
```

12. **(2p)** Skriv en rekursiv funktion `rec_match` som tar två strängar som argument och beräknar antalet positions-matchande tecken mellan två strängar. Positions-matchande tecken definieras här som att samma tecken förekommer på samma position i strängarna.

```
[In] : rec_match('HEJ', 'hej')
[Out]: 0
[In] : rec_match('ab', 'abcdef')
[Out]: 2
[In] : rec_match('hej du', 'hej da') #matchande positioner: 0, 1, 2, 3, 4
[Out]: 5
[In] : rec_match('Hej hopp', 'Hejhopp') #matchande positioner: 0, 1, 2, 6
[Out]: 4
```

13. **(2p)** Antag en uppslagstabell `d` med heltal som både nycklar och värden samt en lista `l` med heltal. Skriv en funktion `show_total(d,l)` som skriver ut summan av värden i `d` för nycklarna i intervallet mellan alla på varandra efterföljande värden i `l`. Om t.ex. `l = [2, 5, 10]` så ska funktionen beräkna summan av värden i `d` för nycklar i intervallen $[2, 5)$ och $[5, 10)$. Funktionen ska returnera en uppslagstabell med tupler av intervallen som nykelvärden och summorna som värden, dvs nyckel/värde-paren ska vara på formen `(a, b): v` där `a` och `b` är ändpunkterna på ett intervall och `v` är värdet som beräknats. *Observera* att ogiltiga intervall (tomma intervall eller "intervall" som bara har en ändpunkt) inte ska inkluderas i uppslagstabellen som returneras.

Notera: det ska vara ett halvöppet intervall, så `b` inkluderas *ej* i intervallet. *Inga* bibliotek är tillåtna, inte ens standardbibliotek, dvs inga **`import`**.

```
[In] : d = {1: 30, 2: 500, 5: 10, 9: 1000, 18: 45, 23: 20}
[In] : l = [1, 5, 18, 10] # [18,10) is not a valid interval and should be ignored
[In] : print(show_total(d, l))
[Out]: {(1, 5): 530, (5, 18): 1010}
[In] : l = [-10, -5]  # interval outside any key in d
[In] : print(show_total(d, l))
[Out]: {(-10, -5): 0}
[In] : l = [20]  # no intervals can be formed
[In] : print(show_total(d, l))
[Out]: None
```

14. (a) **(1p)** Skriv en funktion `h(n,f,k)` som som argument tar ett heltal `n`, en funktion `f` och ett heltal `k`. Funktionen `f` ska ta ett heltal som argument och returnera ett heltal. Funktionen `h` ska applicera funktionen `f` på `n` och returnera resten av resultatet vid heltalsdivision med `k` (d.v.s. modulo `k`). Det betyder att funktionen resulterar i ett heltal i intervallet $[0, k-1]$. Se exempel nedan.

(b) **(1p)** Nedan kod kan inte köras då vi inte definierat `f`. Visa hur du kan använda funktionen `h` med hjälp av **`lambda`** för att implementera exemplen nedan.

```
# om f(x) = x
[In] : h(7, f , 5)
[Out]: 2
[In] : h(312, f , 256)
[Out]: 56

# om f(x) = x^2 + x
[In] : h(7, f , 5)
[Out]: 1
[In] : h(312, f , 256)
[Out]: 120
```

15. (a) **(1p)** Skriv en klass `FilterSet` med en konstruktor som tar ett heltal `k` samt två funktioner `f1` och `f2`. Dessa ska sättas som instansattribut. Klassen `FilterSet` ska även innehålla två instansattribut `s1` och `s2` av typ mängd som sätts till tomma mängden vid instansiering.

(b) **(1p)** I uppgift 14 skrev du en hashfunktion `h(n,f,k)`. Vi ska här använda oss utav den funktionen. Lägg till metoderna `add(n)` och `is_present(n)` till klassen `FilterSet`.

- Metoden `add(n)` ska spara hash-representationen av `n` i `s1` och `s2` med hjälp av `f1` och `f2`. I `s1` ska alltså talet `h(n,f1,k)` sparas och i `s2` ska `h(n,f2,k)` sparas.
- Metoden `is_present(m)` ska returnera `True` om hash-representationen av `m` finns i både `s1` och `s2`.

Notera att denna datastruktur kan returnera felaktiga resultat i `is_present` på grund av hash-krockar. Detta betyder att tal som har samma hash-representation som något annat tal kan returnera `True` även om de inte har lagts till. Se exempelanvändning med `512` nedan.

```
# om f1(x) = x och f2(x) = x^2 + x/2
[In] : B = FilterSet(256, f1, f2)
[In] : B.add(0)
[In] : B.add(124)
[In] : B.add(1)
[In] : B.add(259)
[In] : print(B.is_present(1), B.is_present(259))
[Out]: True True
[In] : print(B.is_present(3), B.is_present(257))
[Out]: False False
[In] : print(B.is_present(512)) # incorrect result due to same
                                # hash value as 0 in both functions
[Out]: True
```