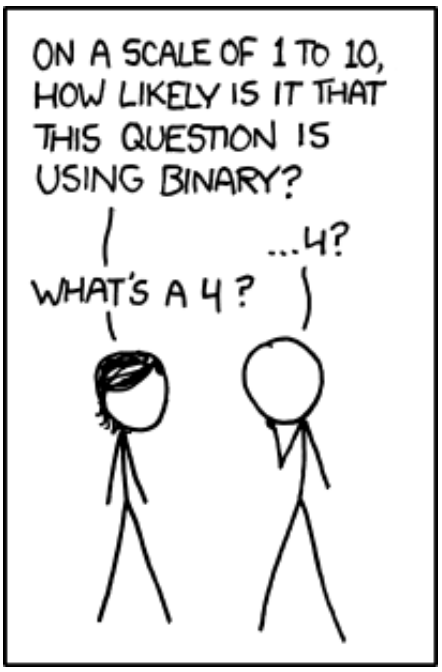
Klausur Informatik I



WS 2013/14

Prof. Nebel

26.02.2014

Aufgabe 1

(Python-Shell, 30 Punkte)

Welche Ausgaben erhalten Sie in der Python-Shell bei Eingabe der folgenden Befehle?

a)

5 P.

```
>>> 'I %s %sam' % ('spam'[2:], 'spam'[:2])
```

b)

5 P.

```
>>> a = ((i, j) for i in range(1, 4) for j in range(1, 7, 2)
...              if i**2 < j*i)
>>> tuple(a)
```

c)

5 P.

```
>>> import re
>>> re.findall(r'[a-z]+?)\w*', "Ham, spam, and, eggs")
```

d)

5 P.

```
>>> from operator import itemgetter
>>> a = [3, 2, 1]
>>> b = ("a", "C", "b")
>>> sorted(zip(a, b), key=itemgetter(1), reverse=True)
```

e)

5 P.

```
>>> a, b = (3, 2, 1, 0), (2, 1, 3, 4, 5)
>>> list(map(max, a, b))
```

f)

5 P.

```
>>> f = lambda y: lambda x: int(x - y/2)
>>> f(-56)(14)
```

Aufgabe 2

(Wissensfragen, 15 Punkte)

a)

5 P.

Welche Berechnungsvorschriften sind Algorithmen? Nennen Sie zur Beantwortung dieser Frage *fünf* Bedingungen, die man an Algorithmen stellen kann, und erläutern Sie diese jeweils kurz.

b)

2 P.

Wodurch unterscheiden sich das *imperative Programmierparadigma* wesentlich vom *deklarativen*?

c)

3 P.

Ordnen Sie die Programmierstile *prozedurale Programmierung*, *funktionale Programmierung*, und *objekt-orientierte Programmierung* jeweils einem der beiden in (b) genannten Grundparadigmen zu. Füllen Sie dazu folgende Tabelle.

```
   Imperativ   |   Deklarativ
_______________|_______________
               |
               |
               |
               |
               |
               |
```

d)

5 P.

Ordnen Sie den in der folgenden Graphik aufgeführten wichtigen Ereignissen in der geschichtlichen Entwicklung der Informatik ihren jeweiligen Jahreszeiten zu.

```
1948        Turing-Maschine
1989        Beginn der Entwicklung des WWW
1971        Erster Mikroprozessor
1941        Erster Speicher-programmierbarer Computer
1936        Erfindung des Z3
```

Aufgabe 3

(Primzahltest, 15 Punkte)

In dieser Aufgabe geht es darum, einen Algorithmus, der eine eingegebene Zahl daraufhin testet, ob sie eine Primzahl ist. Der verwendete Algorithmus *(Sieb des Eratosthenes)* wird dabei durch den folgenden Text in natürlicher Sprache beschrieben.

Das Sieb des Eratosthenes ist ein Verfahren, um für eine natürliche Zahl $n \geq 2$ alle Primzahlen bis zu dieser Grenze zu bestimmen. Zu diesem Zweck erzeugt der Algorithmus zunächst eine Menge aller natürlichen Zahlen von 2 bis (einschließlich) n. Die Menge bezeichnen wir als Sieb. Für jede Zahl i von 2 bis n werden nun, falls i sich noch im Sieb befindet, beginnend mit dem Quadrat von i alle Vielfachen von i, die kleiner oder gleich n sind, aus dem Sieb entfernt. Alle Zahlen, die schließlich im Sieb verbleiben, sind prim.

a)

10 P.

Implementieren Sie diesen Algorithmus in einer Python-Funktion `sieve(n)`, welche für eine natürliche Zahl $n \geq 2$ die Menge aller Primzahlen bis zur Zahl n (also das *Sieb*) zurückgibt.

Hinweis: `s.discard(x)` entfernt `x` aus der Menge `s`, falls diese in der Menge vorhanden ist.

```
def sieve(n):
```

b)

5 P.

Definieren Sie eine Funktion `isprime(n)`, welche für eine natürliche Zahl bestimmt, ob diese eine Primzahl ist. Falls ja, soll die Funktion `True` zurückgeben, andernfalls `False`. Verwenden Sie dazu die Funktion `sieve`. Beachten Sie, dass 0 und 1 keine Primzahlen sind.

```
def isprime(n):
```

Aufgabe 4

(Party, 15 Punkte)

Auf einer Party befinden sich n Gäste, wobei nicht jeder Gast jedem anderen Gast vertraut. Jeder Gast erzählt ausschließlich solchen Gästen geheime Nachrichten, denen er oder sie vertraut.

Um zu überprüfen, ob auf der Party eine geheime Nachricht von einem Gast an einen anderen Gast gelangen kann, könnte man den im folgenden Python-Programm implementierten Algorithmus verwenden. Dabei wird angenommen, dass `confidants` ein Dictionary ist, welches *jedem* Gast der Party eine Liste jener Gäste der Party zuweist, denen er oder sie vertraut. Die Elemente dieser Listen seien jeweils paarweise verschieden.

```
def reachable(sender, receiver, confidants):
    marked = dict((x, False) for x in confidants)
    marked[sender] = True
    L = [sender]
    while L:
       cur = L.pop()
       if cur == receiver:
           return True
         for f in confidants[cur]:
             if not marked[f]:
                 marked[f] = True
                 L.append(f)
    return False
```

a)

P. 7

Wir betrachten den Fall, dass diese Funktion wie folgt aufgerufen wird:

```
cnfd = {1: [2, 3], 2: [5, 4], 3: [3], 4: [6, 3], 5: [6], 6:[5]}
reachable(1, 6, cnfd)
```

Bestimmen Sie den Wert von `L` sowie alle positiv markierten Elemente des Dictionary `marked` *vor* jedem Durchlauf der `while`-Schleife. Was ist der Rückgabewert dieses Funktionsaufrufs?

```
 Durchlauf |     L     | alle x mit marked[x] == True
-------------------------------------------------------
     1     |           |
     2     |           |
           |           |
           |           |
           |           |
           |           |
           |           |
           |           |
```

Rückgabewert:

b)

8 P.

Es sei n die Anzahl der Gäste der Party. Bestimmen Sie die Größenordnung der Laufzeit des Algorithmus (in Landau-Notation) in Abhängigkeit von n (kurze Erläuterung).

Vereinfachend gehen Sie davon aus, dass alle Listen-Operatoren sowie der Zugriff und das Setzen eines Dictionary-Elements in *konstanter Zeit* erfolgen kann.

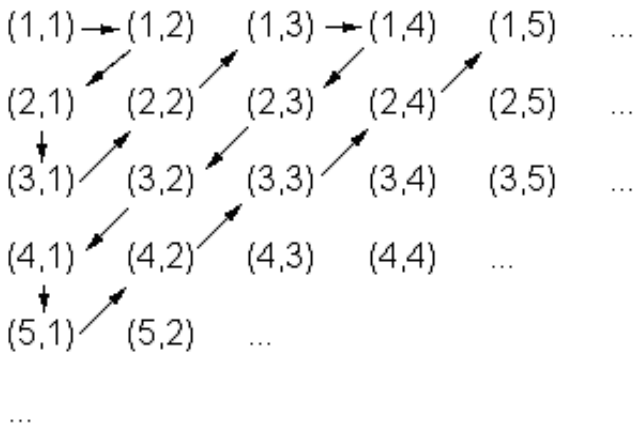
Hinweis: Überlegen Sie sich, wie oft der Ausdruck `f in confidants[cur]` höchstens aufgerufen werden kann.

Aufgabe 5

(Generatoren, 15 Punkte)

Jede positive rationale Zahl lässt sich als ein Bruch zweier von 0 verschiedener natürlicher Zahlen schreiben. Im folgenden stellen wir einen solchen Bruch durch ein Python-Tupel `(zaehler, nenner)` dar. Somit lässt sich z.B. die rationale Zahl $\frac{1}{4}$ durch das Tupel `(1, 4)`, aber auch durch die Tupel `(2, 8)`, `(3, 12)`, etc. repräsentieren (diese Tupel bezeichnen wir daher als *Repräsentanten* der rationalen Zahl).

Man kann nun alle Repräsentanten von positiven rationalen Zahlen sukzessive nach folgenden Schema durchlaufen:



a)

10 P.

Definieren Sie einen Generator `fractions`, der über alle Repräsentanten positiver rationaler Zahlen nach diesem Schema iteriert.

Hinweis: Beachten Sie die Summer der Tupel-Komponenten: gerade / ungerade?

```
def fractions():
```

b)

5 P.

Schreiben Sie eine Funktion, die den in (a) definierten Generator benutzt, um die ersten n Repräsentanten (entsprechend dem obigen Schema) auf der Konsole auszugeben.

```
def print_fractions(n):
```

Aufgabe 6

(Dekoratoren und Namensräume, 10 Punkte)

Betrachten Sie folgenden Quellcode:

```
user = 'Alice'
def decorator(f):
    user = 'Bob'
    def wrapper(*args, **kwargs):
        # global user
        res = '%s %s' % (user, f(*args, **kwargs))
        return res
    return wrapper

@decorator
def func1(str):
    return '%s im hörsaal.' % str

def func2(str):
    return '%s %s ein buch.' % (user, str)

print(func1('schreibt'))
print(func2('liest'))
```

a)

5 P.

Welche Ausgaben erscheinen auf der Konsole, wenn man diesen Code ausführt?

b)

5 P.

Was ändert sich, wenn man in der fünften Zeile das Kommentarzeichen und ein Leerzeichen vor `global user` löscht? Begründen Sie.

Aufgabe 7

(Marmelade, 10 Punkte)

Betrachten Sie das folgende kleine Python-Programm:

```
class Marmelade:
    step = 5

    def __init__(start, limit):
        self.run = start
        self.limit = limit

    def __iter__(self):
        return self

    def __next__(self):
        self.run += step
        if self.run > self.limit:
            raise StopIteration
        return run

jam = Marmelade(32, 45)
for i in jam:
    print(i, end=', ')

for i in jam:
    print(2*i, end=', ')

print("Empty!")
```

a)

4 P.

Dieses Programm ist nicht lauffähig, d.h. es führt zu Laufzeitfehlern. Welche drei Korrekturen müssen Sie vornehmen, damit das Programm fehlerfrei ausgeführt werden kann? Geben Sie die entsprechenden Korrekturen der jeweiligen Zeile oben in der rechten Spalte an.

b)

4 P.

Welche Ausgaben macht das Programm, nachdem Sie die Korrekturen vorgenommen haben?

c)

2 P.

Diese Klasse implementiert ein bestimmtes Protokoll. Wie nennt man dieses Protokoll?

Aufgabe 8

(Funktionen, 10 Punkte)

a)

5 P.

Was berechnet im folgenden Python-Code die Function `afunc`? Vervollständigen Sie dazu den Doc-String an den gekennzeichneten Stellen. Was berechnet die Funktion `bfunc`? Fügen Sie hier einen vollständigen Doc-String ein.

```
def afunc(f, k, n):
    """...


    Arguments:
    f -- A function that expects an integer and returns an integer.
    k -- ...
    n -- ...
    """
    res = 1
    for i in range(k, n+1):
        res *= f(i)
    return res


def bfunc(n):
    """
    ...
    ...
    """
    return afunc(lambda x: 2 * x, 1, n)
```

b)

5 P.

Die Function `cfunc` soll die gleiche Funktionalität wie die Function `afunc` bieten, jedoch rekursiv definiert werden. Vervollständigen Sie dazu folgenden Code-Ausschnitt an den markierten Stellen.

```
def cfunc(f, k, n):
    """Recursive version of function afunc.

    """
    if ________________:
        return 1
    else:
        return ________________ * cfunc(________________)
```