Informatik 5. Klasse

Examensvorbereitung

Teil 1

Python: Aufgabe 1

Notiere jeweils die Ausgaben des folgenden Programmfragments, und erkläre, wie das Resultat zustande kommt.

```
print(7 / 2)

print(7 // 2)

print(7 ** 2)

print(7 % 2)

print(2 % 7)

print(int(2))

print(float(2))

print(bool(2))
```

Python: Aufgabe 2

Notiere jeweils die Ausgaben des folgenden Programmfragments, und erkläre, wie das Resultat zustande kommt.

```
x = 5
y = 1
x = x + y + 3
x = x * 2
print(x)

z = 10
z -= 8
z *= 7
print(z)

a, b, c, d = 4, 7, 1, 6
a, b, c, d = b, c, d, a
a, b, c, d = c, d, a, b
print(a, b, c, d)
```

Python: Aufgabe 3

Notiere jeweils die Ausgaben des folgenden Programmfragments, und erkläre, wie das Resultat zustande kommt.

```
print(True or not False and True)

print(7 < 9 or 7 != 3)

print(5 >= 5 and 1 == 4)

a = 7
if a > 0:
    a = a + 2
a = a + 3
print(a)

b = -1
if b == 0:
    b = b + 2
else:
    b = b + 3
b = b + 4
print(b)

c = 7
if c < 2:
    c = c + 1
elif c < 3:
    c = c + 2
else:
    c = c + 4
print(c)
```

Python: Aufgabe 4

Notiere jeweils die Ausgaben des folgenden Programmfragments, und erkläre, wie das Resultat zustande kommt.

```
for i in range(0, 4):
    print(3*i)

for i in range(2, 8, 3):
    print(i)

for x in [7, 2, 3, 8, 5]:
    if i % 2 == 0:
        break
    print(x)

for x in [7, 2, 3, 8, 5]:
    if i < 4:
        continue
    print(x)
```

Python: Aufgabe 5

Notiere jeweils die Ausgaben des folgenden Programmfragments, und erkläre, wie das Resultat zustande kommt.

```
A = [5, 8, 7, 4, 1, 9]

print(A[2])

print(A[-2])

print(A[2:4])

print(A[:3])

print(A.pop())

print(len(A))

print(A.count(2))

B = [4, 7, 1, 5]
B.insert(2, 3)
print(B)

C = [3, 2, 6]
C.append(C.pop(0))
print(C)
```

Python: Aufgabe 6

(a) Notiere jeweils die Ausgaben des folgenden Programmfragments, und erkläre, wie das Resultat zustande kommt.

```
1  def f(x):
2      3*x - 2
3
4  def g(x, y):
5      z = 1
6      return x + y + z
7
8  def h(L):
9      return L.pop(0)
10
11 def r(x):
12     if x < 3:
13         return x
14     else:
15         return 3 * r(x-2) + 1
16
17 z = 5
18 L = [9, 2, 7]
19
20 print(f(5))
21
22 print(g(3, 4))
23
24 print(h(L))
25
26 print(r(6))
27
28 print(z)
29 print(L)
```

(b) Erkläre den Unterschied zwischen *formalen* und *aktuellen Parametern*.

Python: Aufgabe 7

Notiere jeweils die Ausgaben des folgenden Programmfragments, und erkläre, wie das Resultat zustande kommt.

```
1  print(2 * 'abc' + 'xyz')
2
3  print(len('Hello World!'))
4
5  print(int('10') + float('11'))
6
7  print(ord('A'))
8
9  print(chr(79))
10
11 s = 'Informatik'
12 print(s[2:8])
13
14 print('zb'.upper())
15
16 print('baaba'.replace('b', 'a'))
17
18 datum = '2024-06-07'
19 print('.'.join(datum.split('-')))
20
21 print('ANANAS'.count('A'))
22
23 print('{0} + {1} + {1} = {2}'.format(3, 4, 3 + 2*4))
```

Python: Aufgabe 8

Notiere jeweils die Ausgaben des folgenden Programmfragments, und erkläre, wie das Resultat zustande kommt.

```
BCT = {'A': 1, 'N': 2, 'S': 6}
text = 'ANANAS'
print(BCT[text[-1]])

mydict = dict()
mydict['red'] = 'rot'
mydict['green'] = 'grün'
mydict['blue'] = 'blau'
del mydict['red']
print(len(mydict))

graph = {
    1: [2, 3],
    2: [1, 3],
    3: [1, 2]
}

for v in sorted(graph): # Iteriert über die sortierten Schlüssel
    for w in graph[v]:
        print('{0} -> {1}'.format(v, w))
```

Python: Aufgabe 9

Das folgende Programm soll die Anzahl der ungeraden Zahlen in der Liste `L` bestimmen und auf der Shell ausgeben, enthält aber mehrere Fehler.

```
L = [3, 5, 4  7, 9, 6]

for i in range(1, 7)
    if L[i] % 2 = 0:
        anzahl =+ 1

 print(anzahl)
```

Finde möglichst viele dieser Fehler und gib jeweils an, ob es sich um einen Syntaxfehler, einen Laufzeitfehler oder um einen logischen Fehler handelt.

Python: Aufgabe 10

Schreibe eine Funktion `steuern(E)`, die nach den folgenden Regeln die Steuern aus dem Einkommen `E` berechnet und als Wert der Funktion zurückgibt.

- Ist das Einkommen kleiner als 100 000 Franken, betragen die Steuern 10% des Einkommens.
- Ist das Einkommen kleiner als 200 000 Franken aber nicht kleiner als 100 000 Franken, betragen die Steuern 20% des Einkommens.
- Für alle anderen Einkommen betragen die Steuern 30% des Einkommens.

Python: Aufgabe 11

Scheibe eine Funktion `mittelwert(L)`, welche eine Liste `L` von Zahlen als Argument hat, den Mittelwert dieser Zahlen berechnet und als Wert zurückgibt.

Python: Aufgabe 12

Schreibe eine Funktion `positive(L)`, die aus der Liste von Zahlen `L` eine Liste der positiven Werten von `L` erzeugt und zurückgibt.

Beispiele:

- `positive([1, -4, 3, 7, -1])` ⇒ `[1, 3, 7]`
- `positive([-5, 0, -2])` ⇒ `[]`
- `positive([8, 2, 1, 5])` ⇒ `[8, 2, 1, 5]`

Python: Aufgabe 13

Schreibe eine Funktion `find_item(L, item)`, die in der Liste `L` nach dem ersten Auftreten von `item` sucht und dessen Listenindex zurückgibt, falls es sich in der Liste befindet. Ist die Suche erfolglos, gibt die Funktion `False` zurück.

Beispiele:

- `find_item([1, 3, 5, 7], 5)` ⇒ `2`
- `find_item([1, 3, 5, 7], 4)` ⇒ `False`

Algorithmen: Aufgabe 1

Beschreibe möglichst genau, was ein Algorithmus ist und beurteile damit die Aussage "Ein Kochrezept ist ein Algorithmus".

Algorithmen: Aufgabe 2

Gib in der O-Schreibweise an, in welcher asymptotischen Laufzeitklasse die folgenden Funktionen liegen. $T(n)$ bezeichnet jeweils die Kostenfunktion in Abhängigkeit der Problemgrösse $n \geq 1$.

(a) $T(n) = 2n^2 + 3n - 1$

(b) $T(n) = (n^3 + 4n^2 + n - 2) + (n^2 + 9n + 3)$

(c) $T(n) = (n^3 + 4n^2 + n - 2) \cdot (n^2 + 9n + 3)$

(d) $T(n) = 7.5$

(e) $T(n) = 3^{n+2}$

(f) $T(n) = \log(5n^3)$

Algorithmen: Aufgabe 3

Unten links ist der Quellcode eines Python-Programms abgebildet und unten rechts sein Output bei der Ausführung auf einem Laptop.

```python
from time import time

def my_function(n):
    s = 0
    for i in range(0, n):
        s += i
    return s

for i in range(1, 10):
    n = i * 10**6
    t1 = time()
    my_function(n)
    t2 = time()
    print(n, round(t2-t1, 3), 's')
```

```
1000000  0.056 s
2000000  0.101 s
3000000  0.151 s
4000000  0.214 s
5000000  0.275 s
6000000  0.308 s
7000000  0.377 s
8000000  0.408 s
9000000  0.465 s
```

(a) Beschreibe in groben Zügen, was das Programm macht und was die Zahlen rechts bedeuten.

(b) Welche asymptotische Laufzeitkomplexität hat `my_function`? Begründe die Antwort aus praktischer und theoretischer Perspektive.

(c) Wie können die Unterschiede den theoretischen und den praktischen Ergebnissen erklärt werden?

Algorithmen: Aufgabe 4

(a) Nenne die wichtigsten Komplexitätsklassen für die asymptotische Laufzeit von Algorithmen und gib jeweils ein Beispiel dafür an.

(b) Wo liegt im Allgemeinen die Grenze für die praktische Berechenbarkeit?

Algorithmen: Aufgabe 5

Ein Programm in der Komplexitätsklasse $O\big(n^2\big)$ benötigt für $n = 200$ Eingaben etwa 0.1 Sekunden. Schätze die Dauer, die das Programm unter sonst gleichen Bedingungen für $n = 1000$ Eingaben benötigt.

Algorithmen: Aufgabe 6

Ein Programm in der Komplexitätsklasse $O\big(\log(n)\big)$ für benötigt für $n = 1000$ Eingaben etwa 3 Sekunden. Schätze die Dauer, die das Programm unter sonst gleichen Bedingungen für $n = 10\,000$ Eingaben benötigt.

Algorithmen: Aufgabe 7

Ein Programm in der Komplexitätsklasse $O\big(3^n\big)$ benötigt für $n = 5$ Eingaben etwa 24 Minuten. Schätze die Dauer, die das Programm unter sonst gleichen Bedingungen für $n = 4$ Eingaben benötigt.

Algorithmen: Aufgabe 8

Ein Programm in der Komplexitätsklasse $O\big(n!\big)$ benötigt für $n = 8$ Eingaben etwa 1 Sekunde. Schätze die Dauer, die das Programm unter sonst gleichen Bedingungen für $n = 10$ Eingaben benötigt.

Algorithmen: Aufgabe 9

Ein Programm in der Komplexitätsklasse $O\big(1\big)$ für $n = 2000$ Eingaben etwa 20 Millisekunden. Schätze die Dauer, die das Programm unter sonst gleichen Bedingungen für $n = 3000$ Eingaben benötigt.

Algorithmen: Aufgabe 10

Gib an, in welcher Komplexitätsklasse die folgenden Funktionen liegen, wenn der Parameter `n` jeweils die Problemgrösse bezeichnet.

```python
def funktion_a(n):
    x = 1

    for i in range(0, n):
        x = x + 1

    return x
```

```python
def funktion_b(n):
    x = 1

    for i in range(0, n):
        x = x + i
        for j in range(0, n):
            x = x + j

    return x
```

```python
def funktion_c(n):
    L = [2,3,4,5]
    x = 100
    x = x + 3
    x = x * L[1]
    return x
```

```python
def funktion_d(n):
    x = 1

    for i in range(0, n):
        x = 2 * x

    for j in range(0, 2*n):
        x = x + j

    return x
```

```python
def funktion_e(n):
    x = 0
    i = 1

    while i < n+1:
        x = x + 1
        i = 2*i

    return x
```

Algorithmen: Aufgabe 11

(a) Zeige, wie der klassische euklidsche Algorithmus den grössten gemeinsamen Teiler von $a = 72$ und $b = 40$ berechnet.

(a) Zeige, wie der moderne euklidsche Algorithmus den grössten gemeinsamen Teiler von $a = 72$ und $b = 40$ berechnet.

(c) Warum berechnet die moderne Variante den ggT im Allgemeinen schneller als die klassische Variante?

(d) Für welche Eingaben ist der Aufwand bei beiden Algorithmen etwa gleich gross?

Algorithmen: Aufgabe 12

Implementiere den modernen euklidschen Algorithmus zur Bestimmung des grössten gemeinsamen Teilers von zwei ganzen Zahlen a und b als Python-Funktion.

Halbordnungen: Aufgabe 1

Dies ist kein Aufgabe. Es geht darum, die verwendeten Begriffe zu verdeutlichen.

Eine *Halbordnung* – (Synonyme: *partielle Ordnung*, *Teilordnung*) auf einer Menge M ist eine Relation „$\leq$“, die für alle $x, y, z \in M$ die folgenden drei Eigenschaften hat:

- $x \leq x$
- $x \leq y$ und $y \leq x \;\Rightarrow\; x = y$
- $x \leq y$ und $y \leq z \;\Rightarrow\; x \leq z$

Wichtig ist hier, dass wir nicht verlangen, dass jedes Element mit allen anderen vergleichbar ist sondern nur, dass jedes Element mit sich selber vergleichbar ist.

Beispiel 1: Die Potenzmenge $\mathcal{P}(A)$ der Menge $A = \{1, 2, 3\}$ besteht aus allen Teilmengen der Menge A; also $\mathcal{P}(A) = \big\{\{\,\}, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\big\}$. Auf $\mathcal{P}(A)$ lässt sich eine Halbordnung definieren, indem wir für zwei Mengen $X, Y \in \mathcal{P}(A)$ sagen, dass $X \leq Y$ gilt, wenn X eine Teilemenge von Y ist. Z. B. gilt $\{\,\} \leq \{1, 2\}$ und $\{1, 3\} \leq \{1, 2, 3\}$ aber $\{1, 3\} \not\leq \{1, 2\}$. Beachte, dass die leere Menge Teilmenge *jeder* Menge ist.

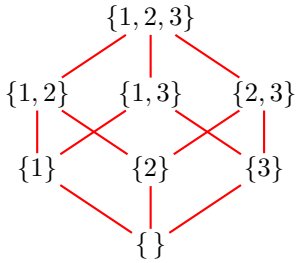
Beispiel 2: Die Teilermenge T_n einer natürlichen Zahl ist die Menge aller natürlichen Zahlen, die n ohne Rest teilen. Etwa $T_{12} = \{1, 2, 3, 4, 6, 12\}$ oder $T_{13} = \{1, 13\}$.

Sind $x, y \in T_n$, so sagen wir, dass $x \leq y$ gilt, wenn x ein Teiler von y ist. In T_{12} gilt dann $1 \leq 4$ oder $2 \leq 6$ aber $3 \not\leq 4$. Hier müssen wir uns daran gewöhnen, dass die Teilbarkeit und nicht die herkömmliche Grössenrelation die Ordnung auf T_n definiert.

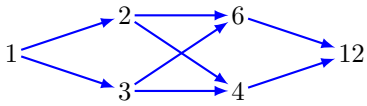
Nun benötigen wir noch die Begriffe des direkten Nachfolgers und des direkten Vorgängers. Es sei $\leq$ eine Halbordnung auf der Menge M und $x, y \in M$. Gilt $x < y$ und gibt es kein Element $z \in M$ mit $x \leq z$ und $z \leq y$, dann ist x ein *direkter Vorgänger* von y und y ein *direkter Nachfolger* von x.

Ist M eine Menge mit einer Halbordnung $\leq$, so stellen wir M durch einen Graphen dar, indem wir jedes Element durch einen Pfeil mit seinem direkten Nachfolger verbinden, sofern es einen solchen hat. Anstelle von Pfeilen kann man auch den direkten Vorgänger jeweils unterhalb seines direkten Nachfolgers zeichnen. Dieser Graph heisst *Hasse-Diagramm*.

Beispiel 3: Das Hasse-Diagramm der Potenzmenge von $A = \{1, 2, 3\}$



Beispiel 4: Hasse-Diagramm von T_{12} mit der Teilbarkeit als Ordnungsrelation.



Halbordnungen: Aufgabe 2

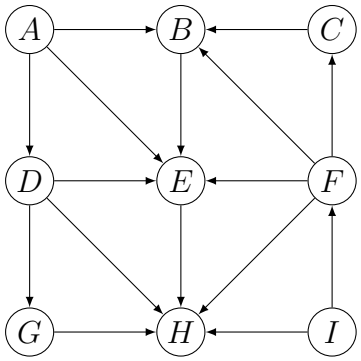
Skizziere das Hasse-Diagramm für T_{30} und der Teilbarkeit als Ordnungsrelation.

Halbordnungen: Aufgabe 3

Stelle die Menge der Mengen
$M = \big\{\{\,\},\{3\},\{4\},\{1,2\},\{2,4\},\{1,2,3\},\{2,3,4\},\{1,2,3,4\}\big\}$
mit der Teilmengenrelation ($\subseteq$) als Ordnung ($\leq$) in einem Hasse-Diagramm dar.

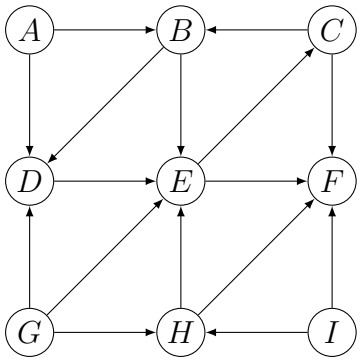
Halbordnungen: Aufgabe 4

Bestimme zum folgenden gerichteten Graphen eine topologische Sortierung oder stelle fest, dass dies nicht möglich ist.



Halbordnungen: Aufgabe 5

Bestimme zum folgenden gerichteten Graphen eine topologische Sortierung oder stelle fest, dass dies nicht möglich ist.



Halbordnungen: Aufgabe 6

Bestimme zum folgenden gerichteten Graphen eine topologische Sortierung oder stelle fest, dass dies nicht möglich ist.