Hilfsmittel

Unicode-Masken

```
Unicode(hex)  Maske
bis 7F        0xxxxxxx
bis 7FF       110xxxxx 10xxxxxx
bis FFFF      1110xxxx 10xxxxxx 10xxxxxx
bis 10FFFF    11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
```

Zeichen und die dezimalen ASCII-Werte

```
chr:      !   "   #   $   %   &   '   (   )   *   +   ,   -   .   /
asc: 32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47
chr:  0   1   2   3   4   5   6   7   8   9   :   ;   <   =   >   ?
asc: 48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63
chr:  @   A   B   C   D   E   F   G   H   I   J   K   L   M   N   O
asc: 64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79
chr:  P   Q   R   S   T   U   V   W   X   Y   Z   [   \   ]   ^   _
asc: 80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95
chr:  `   a   b   c   d   e   f   g   h   i   j   k   l   m   n   o
asc: 96  97  98  99  100 101 102 103 104 105 106 107 108 109 110 111
chr:  p   q   r   s   t   u   v   w   x   y   z   {   |   }   ~
asc: 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127
```

Weitere Zeichen mit ihrem dezimalen Unicode-Codepoint:

```
ä 228 - ö 246 - ü 252 - Ä 196 - Ö 214 - Ü 220 - ß 223 - € 8364
alpha α 945 - beta β 946 - gamma γ 947 - delta δ 948 - epsilon ε 949
pik ♠ 9824 - herz ♡ 9825 - karo ♢ 9826 - kreuz ♣ 9827
```

BOM für UTF-8 Codierung: EFBBBF

1. (3 Punkte) Das Zeichen *Tangut Component-754* hat den hexadezimalen Unicode Code Point `18AF0`. Gib die UTF8 Codierung in hexadezimaler Schreibweise an.

Lösung:

```
Gegebener Codepoint : 18AF0
Binärdarstellung : 0001 1000 1010 1111 0000
           Maske : 1111 0xxx 10xx xxxx 10xx xxxx 10xx xxxx
      eingesetzt : 1111 0000 1001 1000 1010 1011 1011 0000
  UTF-Codierung : F098ABB0
```

2. (3 Punkte) Ein Texteditor stellt folgende Bitfolge dar:

```
1100 0011 1011 0110 0100 0001 1100 1110 1011 0011 0111 0111
```

Gib die hexadezimalen Codepoints der dargestellten Zeichen an. Prüfe, ob die Zeichen in der Hilfsmitteltabelle auftauchen und gib dann an, um welche Zeichen es sich handelt.

Lösung:

```
1100 0011 1011 0110 0100 0001 1100 1110 1011 0011 0111 0111
Die Bitfolge enthält 4 Zeichen:
Zeichen 1:  codepoint dezimal = 246, codepoint hex = f6, Zeichen ö
Zeichen 2:  codepoint dezimal = 65, codepoint hex = 41, Zeichen A
Zeichen 3:  codepoint dezimal = 947, codepoint hex = 3b3, Zeichen gamma
Zeichen 4:  codepoint dezimal = 119, codepoint hex = 77, Zeichen w
```

3. (3 Punkte) Ein Texteditor stellt folgende Bitfolge dar:

```
1110 1111 1011 1011 1011 1111 0111 0100 1110 1010 1001 1010 1010 0010 0011 1001
```

Gib die hexadezimalen Codepoints der dargestellten Zeichen an. Prüfe, ob die Zeichen in der Hilfsmitteltabelle auftauchen und gib dann an, um welche Zeichen es sich handelt.

Lösung:

```
1110 1111 1011 1011 1011 1111 0111 0100 1110 1010 1001 1010 1010 0010 0011 1001
Die Bitfolge enthält nach dem BOM 3 Zeichen:
Zeichen 1:  codepoint dezimal = 116, codepoint hex = 74, Zeichen t
Zeichen 2:  codepoint dezimal = 42658, codepoint hex = a6a2, Zeichen nicht in Hilfstabelle
Zeichen 3:  codepoint dezimal = 57, codepoint hex = 39, Zeichen 9
```

4. (3 Punkte) Decodiere die Bitfolge mit dem angegebenen Huffman-Baum. Gib für jedes Zeichen die Codierung an.

```
110100111111000111000011101
```

Hinweis zur Darstellung des Huffman-Baums: Der Baum ergibt sich durch Drehen der abgebildeten Zeilen um 90 Grad nach rechts. Die Punkte geben die Ebenen des Baumes an. Knoten, die nur Zahlen aber keine Zeichen enthalten, sind mit einem # markiert.

```
...2 t
..4 #
...2 b
.6 #
...1 g
..2 #
...1 u
10 #
..2 r
.4 #
..2 e
```

Lösung:

```
Codierung der Zeichen: e -> 00, r -> 01, u -> 100, g -> 101, b -> 110, t -> 111
butterberg
```

5. (3 Punkte) Konstruiere nach dem Verfahren aus dem Unterricht einen Huffman-Baum für das Wort `gegengerade`. Gib für jeden Buchstaben die Codierung an.

Lösung:

```
Huffmanbaum für: gegengerade
..4 e
.7 #
..3 g
11 #
...1 d
..2 #
...1 a
.4 #
...1 r
..2 #
...1 n

Codierung:
g -> 10
e -> 11
n -> 000
r -> 001
a -> 010
d -> 011
```

6. (3 Punkte) a. Was sagt die Fano-Bedinung aus?

   b. Unten sind Codewörter für einige Buchstaben gegeben. Ist die Fano-Bedingung erfüllt? Falls nein, erläutere die Aussage der Fano-Bedingung mit einem Beispiel.

   a = '00' b = '10' c = '11' d = '01' f = '111' g = '100'

Lösung: a. Die Fanobedingung besagt, dass kein Codewort der Anfang eines anderen Codewortes sein darf. Wenn die Fanobedingung erfüllt ist, lassen sich Codierungen eindeutig decodieren.

b. Das Codewort für c ist der Anfang des Codeworts für f. Die Fano-Bedingung ist nicht erfüllt: Beispiel: cbca und fdg haben beide die Codierung 11101100.

7. (3 Punkte) Es sei $f(n)$ die Laufzeit eines Algorithmus in Abhängigkeit von der Eingabe n. Bestimme die zugehörige Komplexitätsklasse.

   a) $f(n) = n^2 + 2n \cdot \log n$
   b) $f(n) = 3n^2 \cdot \log n + 0.5 \cdot 2^n$
   c) $f(n) = 1000$
   d) $f(n) = 5 + 6 + ... + n$
   e) $f(n) = \log(n) + n$

Lösung: a) $f(n) \in O(n^2)$ b) $f(n) \in O(2^n)$ c) $f(n) \in O(1)$ d) $f(n) \in O(n^2)$ e) $f(n) \in O(n)$

8. (2 Punkte) In welcher Komplexitätsklasse ist die Laufzeit der Funktion `foo` in Abhängigkeit von der Länge n der übergebenen Liste a im best case und im worst case?

```
def foo(a):
    n = len(a)
    for i in range(n):
        if a[i] == 7:
            for j in range(n):
                for k in range(n):
                    print("hello")
```

Lösung: best case: $O(n)$, worst case: $O(n^3)$

9. (2 Punkte) In welcher Komplexitätsklasse ist die Laufzeit der Funktion `foo` in Abhängigkeit von der Länge n der übergebenen Liste a im best case und im worst case?

```
def foo(a):
    n = len(a)
    for i in range(n):
        if a[i] == 42:
            return True
    return False
```

Lösung: best case: $O(1)$, worst case: $O(n)$

10. (3 Punkte) Sortiere die Zahlenfolge mit SelectionSort. Schreibe für die ersten drei Durchgänge je eine Zeile.

```
88 14 3 16 20 42 9
```

Lösung:

```
3 14 88 16 20 42 9
3 9 88 16 20 42 14
3 9 14 16 20 42 88
```

11. (2 Punkte) In unserer Implementierung des SelectionSort-Algorithmus wurden Variablennamen geändert. Dabei haben sich Fehler eingeschlichen. Korrigiere die fehlerhaften Stellen.

```
def selection_sort(hugo):
    for kai in range(len(hugo)-1):
        otto = kai
        lena = hugo[kai]
        for j in range(kai+1,len(hugo)-1):
            if hugo[j] < lena:
                otto = j
                kai = hugo[j]
        hugo[otto],hugo[kai] = hugo[kai],hugo[otto]
```

Lösung:

```
In Zeile 5: for j in range(kai+1,len(hugo))
In Zeile 8: lena = hugo[j]
```

12. (3 Punkte) Sortiere die Zahlenfolge mit BubbleSort. Schreibe für die ersten drei Durchgänge je eine Zeile.

```
23 10 4 66 42 12 17
```

Lösung:

```
10 4 23 42 12 17 66
4 10 23 12 17 42 66
4 10 12 17 23 42 66
```

13. (3 Punkte) In unserer Implementierung des Bubblesort-Algorithmus wurde ein print-Statement eingefügt. Welche Zahlenfolge wird bei dem angegebenen Aufruf von Bubblesort ausgegeben? (Der print-Parameter end=' ' sorgt dafür, dass die Zahlen nicht in einer neuen Zeile, sondern mit blank getrennt ausgegeben werden.)

```
def bubble_sort(a):
    getauscht = True
    while getauscht:
        getauscht = False
        for i in range(len(a)-1):
            if a[i] > a[i+1]:
                print(a[i],end=' ')
                a[i],a[i+1]=a[i+1],a[i]
                getauscht = True

bubble_sort([5,2,9,1,6,6,3])
```

Lösung:

```
5 9 9 9 9 5 6 2 6 5
```

14. (2 Punkte) Welche Komplexität hat die Laufzeit von BubbleSort im best, worst und average-case? In welcher Komplexitätsklasse ist der zusätzliche Platzbedarf?

Lösung:

best: $O(n)$, average: $O(n^2)$, worst: $O(n^2)$, Platz: $O(1)$

Aufgabe:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Summe:
Punkte:	3	3	3	3	3	3	3	2	2	3	2	3	3	2	38