Aufgabe 1

`W=0x57`, `o=0x6F`, `w=0x77`, `!=0x21`

(a) UTF-8: `57 6F 77 21`

(b) UTF-16: `0057 006F 0077 0021`

(c) UTF-32: `00000057 0000006F 00000077 00000021`

Aufgabe 2

ef	b7	b2
1110 1111	10 110111	10 110010

Ja, denn das erste Byte zeigt mit der Startsequenz `1110` an, dass es sich um eine 3-Byte-Folge handelt. Die folgenden beiden Bytes beginnen jeweils mit der Sequenz `10`, was ebenfalls korrekt ist.

Aufgabe 3

F	o	r	g	e	t		i	t	!
46	6f	72	67	65	74	20	69	74	21
襯		牧		整		n. def.		瑠	

Natürlich muss man die Zeichen in der untersten Zeile weder kennen noch angeben. Wer nach einem bestimmten Unicode-Zeichen sucht, findet die nötigen Informationen unter `www.unicode.org/charts/`.

Aufgabe 4

27	01
00100111	00000001

(a) UTF-16: `00100111 00000001`

(b) UTF-32: `00000000 00000000 00100111 00000001`

(c) UTF-8: Platzbedarf ermitteln:

1 Byte	2 Byte	3 Byte	4 Byte
0xxxxxxx	110xxxxx	1110xxxx	11110xxx
	10xxxxxx	10xxxxxx	10xxxxxx
		10xxxxxx	10xxxxxx
			10xxxxxx
0-7 Bit	8-11 Bit	12-16 Bit	17-21

Die Zeichennummer besteht aus 14 Bit. Also sind für UTF-8 drei Bytes nötig: `1110`0010 `10`011100 `10`000001.

Aufgabe 5

(a) Nein, das erste Byte beginnt mit einer 1 ⇒ ISO-8859-1

(b) Ja, die Zeichenfolge ist 2 Byte = 16 Bit lang.

(c) Ja, die Zeichenfolge besteht aus zwei ASCII-Zeichen.

(d) Nein, das erste Byte zeigt an, dass das UTF-8-Zeichen aus zwei Bytes besteht. Dann müsste das zweite Byte die Form `10xxxxxx` haben.

Aufgabe 6

(a) hexadezimal ⇒ binär:

F0	93	81	A8
1111000	10010011	10000001	10101000

Die Bits an den unterstrichenen Stellen codieren das Zeichen

`00 010011 000001 101000`

Teilt man von rechts in Gruppen von 4 Bits auf, so erhält man

0001	0011	0000	0110	1000
1	3	0	6	8

(b) UTF-8-Darstellung mit drei Bytes:

`1110xxxx 10xxxxxx 10xxxxxx`

Also könnten Zeichen bis zu 4 + 6 + 6 = 16 Bit damit codiert werden.

Die bei (a) bestimmte Bitfolge `0001 0011 0000 0110 1000` besteht jedoch aus 17 Bit (die drei führenden Nullen kann man weglassen). Deshalb ist eine Codierung mit weniger als 4 Byte unmöglich.