Übung 2 – Lösung

(Für Aufgaben 1 und 2)

Aufgabe 1)

a)
Initialisierung des Wörterbuchs mit den Zeichen des Zeichenvorrats:

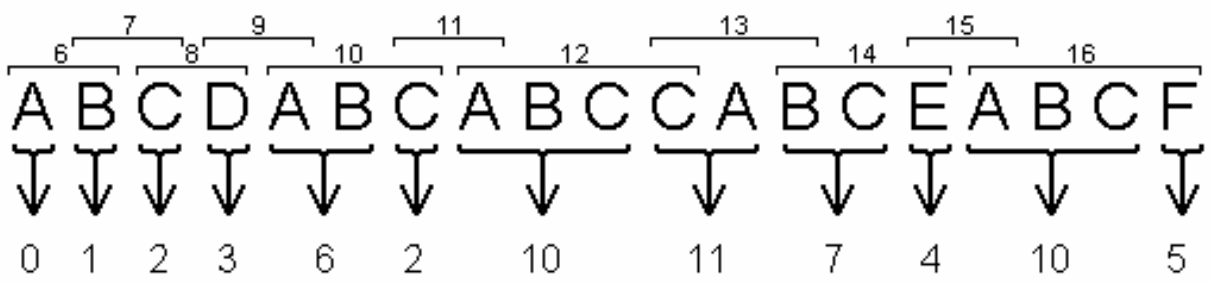
Index	0	1	2	3	4	5
Zeichen	A	B	C	D	E	F

Tabelle zur Darstellung der Schritte des Algorithmus:

k	**p & k**	**neuer Tab-Eintrag**	**Ausgabe**	**p**
				A
B	AB	AB, 6	0	B
C	BC	BC, 7	1	C
D	CD	CD, 8	2	D
A	DA	DA, 9	3	A
B	AB	-	-	AB
C	ABC	ABC, 10	6	C
A	CA	CA, 11	2	A
B	AB	-	-	AB
C	ABC	-	-	ABC
C	ABCC	ABCC, 12	10	C
A	CA	-	-	CA
B	CAB	CAB, 13	11	B
C	BC	-	-	BC
E	BCE	BCE, 14	7	E
A	EA	EA, 15	4	A
B	AB	-	-	AB
C	ABC	-	-	ABC
F	ABCF	ABCF, 16	10	F
EOF	F		5	

⇨ komprimierte Nachricht: 0 1 2 3 6 2 10 11 7 4 10 5

Veranschaulichung:
Die eckigen Klammern in der oberhalb der Nachricht markieren den neuen Eintrag ins Wörterbuch, die runden Klammern unterhalb der Nachricht markieren die aktuelle Ausgabe.



12 Zahlen statt 19 Zeichen

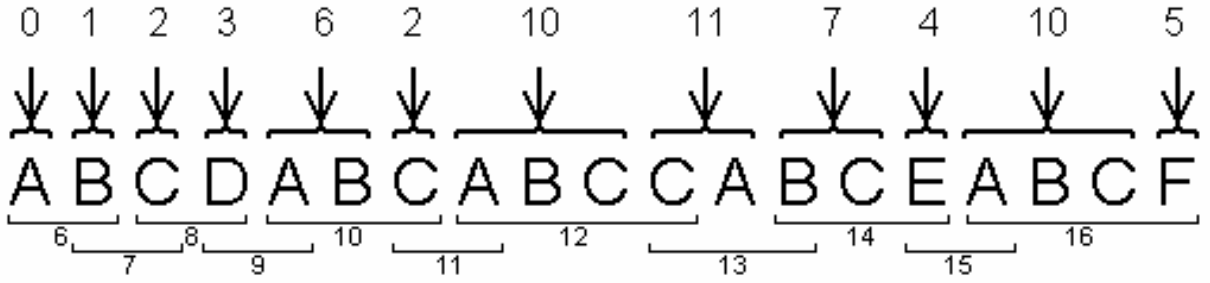
b)
Initialisierung des Wörterbuchs wie bei a)

Tabelle zur Darstellung der Schritte des Algorithmus:

k	**p**	**Ausgabe**	**q**	**neuer Tab-Eintrag**	**old**
0		A			A
1	B	B	AB	AB, 6	B
2	C	C	BC	BC, 7	C
3	D	D	CD	CD, 8	D
6	AB	AB	DA	DA, 9	AB
2	C	C	ABC	ABC, 10	C
10	ABC	ABC	CA	CA, 11	ABC
11	CA	CA	ABCC	ABCC, 12	CA
7	BC	BC	CAB	CAB, 13	BC
4	E	E	BCE	BCE, 14	E
10	ABC	ABC	EA	EA, 15	ABC
5	F	F	ABCF	ABCF, 16	F

- Das Wörterbuch wird auch bei der Dekodierung dynamisch aufgebaut und muss nicht mit übertragen werden.
- Die Ausgabe entspricht der ursprünglichen Nachricht.

Veranschaulichung:



Aufgabe 2)

a)

k	p & k	neuer Tab-Eintrag	Ausgabe	p
				A
B	AB	AB, 6	0	B
C	BC	BC, 7	1	C
D	CD	CD, 8	2	D
A	DA	DA, 9	3	A
B	AB	-	-	AB
A	ABA	ABA, 10	6	A
B	AB	-	-	AB
A	ABA	-	-	ABA
E	ABAE	ABAE, 11	10	E
E	EE	EE, 12	4	E
EOF			4	

b)

k	p	Ausgabe	q	neuer Tab-Eintrag	old
0		A			A
1	B	B	AB	AB, 6	B
2	C	C	BC	BC, 7	C
3	D	D	CD	CD, 8	D
6	AB	AB	DA	DA, 9	AB
10	???				

Veranschaulichung analog Aufgabe 1)

Problem:

- Die Referenz kann nicht aufgelöst werden, da der zugehörige Wörterbucheintrag gerade erst erstellt wird.

c)

I.) Warum kann eine Referenz entstehen, die nicht auflösbar ist?

- Algorithmus zur Kodierung: Erstellt einen Wörterbucheintrag und gibt den ersten Teil davon aus.
- Algorithmus zur Dekodierung: Benötigt beide Teile des nächsten Wörterbucheintrages, um diesen erstellen zu können
- => insgesamt: Algorithmus zur Dekodierung ist mit der Erstellung des Wörterbuchs genau einen Schritt „hinterher“(siehe auch die Veranschaulichung bei Aufgabe 1))
- => Das Problem tritt immer genau dann auf, wenn der Algorithmus zur Kodierung einen Wörterbucheintrag referenziert, den er eben erst (d. h. im direkt vorangegangenen Schleifendurchlauf) erstellt hat.

II.) Welche Zeichenfolgen verursachen das LZW-Problem?

Die Wörterbucheinträge und die Referenzen beim LZW-Algorithmus hängen von der zu kodierenden Nachricht ab. Folglich ist auch von der Nachricht abhängig, ob der unter I.) identifizierte Fall eintritt oder nicht.
Es stellt sich also folgende Frage: Welche Zeichenfolgen verursachen, dass bei der Kodierung der zuletzt erstellte Wörterbucheintrag referenziert wird?

Um diese Frage zu lösen, soll der Algorithmus allgemein für eine beliebige Nachricht betrachtet werden.

Sei A ein beliebige Zeichenvorrat.
Sei $x_1 \ldots x_i \ldots x_j \ldots x_n$ die zu kodierende Zeichenkette (mit $x_k \in A$ für $k = 1,\ldots,n$)

Betrachtet wird ein beliebiger Zeitpunkt während der Kodierung. Annahmen:

1. Der nächste Wörterbucheintrag beginnt mit x_1 (d. h. $p = x_1$)
2. Im Wörterbuch befindet sich bereits die Zeichenfolge $x_1 \ldots x_i$ (wobei wegen der Initialisierung des Wörterbuchs stets gilt: $i <= 1$)

Der Algorithmus zur Kodierung arbeitet demnach wie folgt:
(c_1, c_2, c_3, c_4 seien natürliche Zahlen, die jeweils einen Index aus dem Wörterbuch repräsentieren)

k	**p & k**	**neuer Tab-Eintrag**	**Ausgabe**	**p**
				x_1
x_2	x_1x_2	-	-	x_1x_2
x_3	$x_1x_2x_3$	-	-	$x_1x_2x_3$
…	…	...	...	…
x_i	$x_1 \ldots x_i$	-	-	$x_1 \ldots x_i$
x_{i+1}	$x_1 \ldots x_{i+1}$	$x_1 \ldots x_{i+1}$, c_1	$c_2 = \text{index}(x_1 \ldots x_i)$	x_{i+1}
x_{i+2}	$x_{i+1}\ x_{i+2}$	…	…	…
…	…	…	…	…
x_j	$x_{i+1} \ldots x_j$	$x_{i+1} \ldots x_j$, c_3	$c_4 = \text{index}(x_{i+1} \ldots x_{j-1})$	x_j
…				

Der gerade zuletzt erstellte Wörterbucheintrag würde genau dann referenziert, falls $c_4 = c_1$

- ⇨ LZW-Problem tritt auf, falls $c_4 = c_1$
- ⇨ $x_{i+1} \ldots x_{j-1} = x_1 \ldots x_{i+1}$
- ⇨ $x_{i+1} = x_1$ und $x_{i+2} = x_2$ und … und $x_{j-1} = x_{i+1} = x_1$
- ⇨ Die Ausgangszeichenkette muss die Form $\mathbf{x_1 \ldots x_i\ x_1 \ldots x_i x_1 x_j}$ haben und es gelten die oben gemachten Annahmen 1) und 2)

Das LZW-Problem wird also durch Zeichenfolgen der Form

$$x_1 \ldots x_i \quad x_1 \ldots x_i \quad x_1$$

ausgelöst. Diese Zeichenkette kann alternativ auch dargestellt werden als:

$$Wort_1 \quad Wort_1 \quad 1.ZeichenVonWort_1$$

oder auch:

$$zW \quad zW \quad z$$

wobei z ein einzelnes Zeichen und W eine Zeichenkette beliebiger Länge (auch der Länge 0) sind.
(Probe: durch Dekodierung der allgemeinen Form)

Zusammenfassend:
Sei c ein einzelnes Zeichen und W ein Wort beliebiger Länge (auch der Länge 0). Es tritt genau dann der LZW-Fehler auf, falls

1. Die Zeichenkette **cWcWc** auftritt
2. cW bereits im Wörterbuch enthalten ist und
3. der Algorithmus am Anfang dieser Zeichenkette mit einem neuen Wörterbucheintrag beginnt (d. h. die Zeichenkette nicht z. B. in der Mitte auseinander gerissen wird)

d)
Algorithmus fehlt erstes Zeichen des folgenden Wörterbucheintrages (siehe b) bzw. Veranschaulichung). Dieses ist jedoch vorhersagbar (siehe c)): $x_{i+1} = x_1$.
=> Ausnahmebehandlung: falls Referenz auf unvollständigen Wörterbucheintrag zeigt, dann Wörterbucheintrag vervollständigen, indem das vorderste Zeichen hinten angehängt wird.

```
k := input.getCode();
SeqChar old := tab.get(k);
output.write(old);
SeqChar p := <>;
k := input.getCode();
while k ≠ EOF do
        if (tab.contains(k)) do
                p := tab.get(k);
        else
                p := old & < firstChar(old) >
        output.write(p);
        SeqChar q := old & < firstChar(p) >;
        int c = tab.newCode(q); tab.put(q, c);
        old := p;
        k := input.getCode();
enddo;
```