

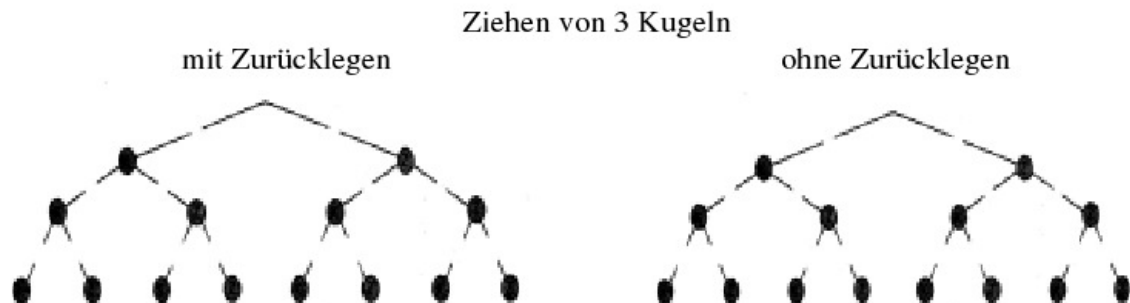
6. Anhang

6.1 Bernoulli-Kette

Urne mit 12 Kugeln: 7 blaue, 5 weiße (andere)

Ziehen von 3 Kugeln

$P(\text{„k Kugeln blau“}) = ? ; k = 0, 1, 2, 3$



Gemeinsamkeiten?

1. 3-stufiges Zufallsexperiment
2. zwei mögliche Ergebnisse auf jeder Stufe

Was ändert sich, wenn die 5 weißen gegen 2 gelbe und 3 grüne Kugeln ausgetauscht werden?

Ein Zufallsversuch mit nur zwei möglichen Ergebnissen (Erfolg bzw. Misserfolg) heißt ein **Bernoulli – Versuch**.

(Jacob Bernoulli 1654-1705)

Wird ein Bernoulli – Versuch n-mal so wiederholt, dass sich die Erfolgswahrscheinlichkeit p und die Wahrscheinlichkeit für einen Misserfolg q ($q = 1-p$) nicht ändert, so spricht man von einer **Bernoullikette**.

(äquivalent: die einzelnen Versuche müssen voneinander unabhängig sein.; siehe Bsp. mit Zurücklegen)

Ergebnisse: 1...Erfolg (blaue Kugel gezogen), 0...Misserfolg (gezogene Kugel ist nicht blau)

k	mögliche Ergebnisse	P("k Kugeln blau")	
		mit Zurücklegen (Bernoullikette)	ohne Zurücklegen
0	000		
1	001		
	010		
	100		
2	011		
	101		
	110		
3	111		

Allgemein:

n-fache Bernoullikette:

1. Welche Wahrscheinlichkeit hat ein Pfad mit genau k Erfolgen ($k = 0, 1, \dots, n$)?

$$P(\text{ein Pfad hat genau 0 Erfolge}) = q^n$$

$$P(\text{ein Pfad hat genau 1 Erfolge}) = p \cdot q^{n-1}$$

$$P(\text{ein Pfad hat genau 8 Erfolge}) = p^8 q^{n-8}$$

$$P(\text{ein Pfad hat genau n Erfolge}) = p^n$$

$$\mathbf{P(\text{ein Pfad hat genau k Erfolge}) = } p^k q^{n-k}$$

2. Wie viele Pfade mit genau k Erfolgen gibt es ($k = 0, 1, \dots, n$)?

Modell: Urne mit n Kugeln,
Ziehen von k Kugeln ungeordnet, ohne Zurücklegen
Wie viele Möglichkeiten gibt es k Kugeln aus n zu ziehen?

Es gibt $\binom{n}{k}$ Möglichkeiten aus n Stufen k Stufen mit Erfolg auszuwählen.

3. Mit welcher Wahrscheinlichkeit treten genau k Erfolge auf ($k = 0, 1, \dots, n$)?

$$P(X = 0) = P(\text{genau 0 Erfolge}) = \binom{n}{0} p^0 \cdot (1-p)^n = \binom{n}{0} p^0 \cdot q^n$$

$$P(X = k) = P(\text{genau k Erfolge}) = \binom{n}{k} p^k \cdot (1-p)^{n-k} = \binom{n}{k} p^k \cdot q^{n-k}$$

In einer Bernoullikette der Länge n mit der Erfolgswahrscheinlichkeit p treten genau k Erfolge mit der Wahrscheinlichkeit

$$P(\text{genau k Erfolge}) = \binom{n}{k} p^k \cdot (1-p)^{n-k}, \quad k = 0, 1, 2, \dots, n \text{ auf.}$$

Anlage 1 zur Prüfungsrichtlinie

Theoretische Prüfung von Bewerbern um eine Fahrerlaubnis/ Prüfbescheinigung

1 Allgemeine Hinweise

1.1 Gegenstand der theoretischen Prüfung ist der im Verkehrsblatt veröffentlichte Fragenkatalog in der jeweils gültigen Fassung. Der Katalog ist gegliedert in

- Teil 1 Grundstoff
- Teil 2 Zusatzstoff.

Der Teil 1 "Grundstoff" stellt den Abschnitt des Fragenkatalogs dar, aus dem unterschiedslos bei allen Prüfungen um eine Fahrerlaubnis Fragen zur Anwendung kommen. Die Fragen des Grundstoffs sind abschnitts- bzw. kapitelweise nummeriert und mit "G" gekennzeichnet.

Etwa die Hälfte der Fragen des Grundstoffs wird auch bei Prüfungen von Mofafahrern eingesetzt. Diese Fragen sind zusätzlich mit "Mofa" gekennzeichnet.

Der Teil 2 "Zusatzstoff" stellt den Abschnitt des Fragenkatalogs dar, aus dem klassenspezifisch - zusätzlich zum Grundstoff - Fragen zur Anwendung kommen. Die Fragen des Zusatzstoffs sind ebenfalls abschnitts- bzw. kapitelweise nummeriert und mit den Kennzeichen der einzelnen Klassen (z. B. B = Klasse B) versehen.

Jede Frage erscheint im Fragenkatalog nur einmal. Die Nummerierung ist so angelegt, dass Fragen für bestimmte Fahrerlaubnisklassen möglichst zusammenstehen. Soweit Fragen einer Nummer des Kapitels A „Prüfung der Kenntnisse“ des Anhangs II der Richtlinie 2000/56/EG zugeordnet sind, werden sie mit der entsprechenden Nummer des Kapitels A gekennzeichnet.

1.2 Wenn in einer Frage oder einer Antwort der Begriff "Fahrzeug" ohne nähere Angaben verwendet wird, ist darunter immer ein Fahrzeug derjenigen Klasse zu verstehen, für die der Bewerber eine Fahrerlaubnis/Prüfbescheinigung beantragt hat.

Fotos oder Grafiken geben die Situation aus der Sicht eines Pkw-Fahrers wieder und finden sinngemäß für alle Bewerber um eine Fahrerlaubnis/Prüfbescheinigung Anwendung.

Auf Fotos erkennbares Fehlverhalten anderer Verkehrsteilnehmer, das sich nicht auf die Frage bezieht, ist für die Beantwortung der Frage ohne Bedeutung.

2 Struktur und Beantwortung der Fragen

2.1 Jede Frage hat höchstens drei Antworten, von denen mindestens eine richtig ist. Die Antworten beziehen sich nicht aufeinander, sondern jeweils nur auf die Frage. Sie stellen lediglich eine Auswahl der zur jeweiligen Frage möglichen Antworten dar.

Die Fragen werden entsprechend ihrer Bedeutung mit 2 bis 5 Punkten bewertet. Die Wertigkeit ist im Katalog bei jeder Frage angegeben.

Im Katalog sind die richtigen Antworten mit "x", die falschen mit "o" gekennzeichnet; die richtigen Antworten sind zuerst aufgeführt. In der Prüfung ist die Reihenfolge der Antworten

beliebig. Bei hinzuschreibenden Antworten ist im Katalog die richtige Zahl in (()) angegeben.

2.2 In der Prüfung sind die Fragen durch

- Ankreuzen von Antworten

oder

- Einsetzen von Zahlen

zu beantworten.

Eine Frage gilt als falsch beantwortet,

- wenn nicht jede richtige Antwort angekreuzt ist,
- wenn eine falsche Antwort angekreuzt ist oder
- wenn eine hinzuschreibende Zahl nicht oder eine falsche Zahl eingetragen ist.

Die Bewertung falsch beantworteter Fragen erfolgt nach Nr. 2.1, Absatz 2, Satz 1. Eine Prüfung ist bestanden, wenn die in Nr. 3.2 bis 3.7.2 jeweils aufgeführte zulässige Fehlerpunktzahl nicht überschritten wird.

3 Zusammenstellung und Bewertung der Prüfungsfragen

3.1 Allgemeine Hinweise

Unter Nr. 3.2 bis 3.7.2 werden Zusammenstellung und Bewertung der Prüfungsfragen für die einzelnen Klassen beschrieben.

Im Grundstoff darf die Zahl der Punkte aus den einzelnen Stoffgebieten um bis zu 4 Punkte gegenüber den Angaben in der Tabelle zu Nr. 3.2.1 (Ersterwerb) und bis zu 2 Punkte gegenüber den Angaben in der Tabelle zu Nr. 3.2.2 (Erweiterung) abweichen, vorausgesetzt, die Summe der Punkte bleibt unverändert.

Bei Erweiterungsprüfungen¹⁾ wird der Grundstoff unter Nr. 3.2.2 erneut mitgeprüft. Bei gleichzeitiger Prüfung mehrerer Klassen in einem Termin wird der Grundstoff jedoch nur einmal geprüft (siehe Nr. 3.8, Beispiel 3 und 4).

Bei der Ermittlung des Prüfungsergebnisses werden die im Grundstoff und die im Zusatzstoff angefallenen Fehlerpunkte zusammen bewertet (siehe Nr. 3.2.1 bis 3.7.2). Dies gilt auch bei Prüfungen von mehreren Klassen in einem Termin. Es gilt auch dann, wenn die Klasse B als notwendige Vorbesitzklasse erstmalig erworben wird.

Die theoretische Prüfung ist nicht bestanden, wenn die in den Tabellen 3.2 bis 3.7 aufgeführten zulässigen Fehlerpunkte überschritten oder zwei Fragen mit Wertigkeit 5 falsch beantwortet werden.

Die Bewertung der Prüfung ist den Tabellen 3.6, 3.7.1 und 3.7.2 zu entnehmen.

¹⁾ Um eine Erweiterungsprüfung handelt es sich immer dann, wenn eine bestandene, noch gültige theoretische Fahrerlaubnisprüfung oder eine Fahrerlaubnis vorhanden ist.

3.2 Zusammenstellung und Bewertung der Fragen für die Fahrerlaubnisklassen A, A1, B, M, S, L und T

3.2.1 Ersterwerb

Stoffgebiet	Abschnitt im Fragenkatalog	Zahl der Fragen	Summe der Punkte
1. Grundstoff			
Gefahrenlehre	1.1	8 (davon 4 Bildfragen)	32
Verhalten im Straßenverkehr	1.2	6 (davon 1 Bildfrage)	21
Vorfahrt/ Vorrang	1.3	3 (mindestens 2 Bildfragen)	15
Verkehrszeichen	1.4	2 (davon mind. 1 Bildfrage)	6
Umweltschutz	1.5	1	3
Summe Grundstoff		20	77
2. Zusatzstoff	2.1 bis 2.8	10	33
Gesamtstoff		30	110
Zulässige Fehlerpunkte 10; es sei denn, zwei Fragen mit Wertigkeit 5 falsch beantwortet.			

3.2.2 Erweiterung

Stoffgebiet	Abschnitt im Fragenkatalog	Zahl der Fragen	Summe der Punkte
1. Grundstoff			
Gefahrenlehre	1.1	4 (davon 2 Bildfragen)	16
Verhalten im Straßenverkehr	1.2	3 (davon 1 Bildfrage)	10
Vorfahrt/Vorrang	1.3	2	10
Verkehrszeichen	1.4	1	3
Summe Grundstoff		10	39
2. Zusatzstoff	2.1 bis 2.8	10	33
Gesamtstoff		20	72
Zulässige Fehlerpunkte 6.			

3.3 Zusammenstellung und Bewertung der Fragen für die Fahrerlaubnisklassen C und C1

3.3.1 Klasse C

Stoffgebiet	Abschnitt im Fragenkatalog	Zahl der Fragen	Summe der Punkte
1. Grundstoff	(wie unter 3.2.2)	10	39
2. Zusatzstoff	2.1 bis 2.8	27	89
Gesamtstoff		37	128
Zulässige Fehlerpunkte 10; es sei denn, zwei Fragen mit Wertigkeit 5 falsch beantwortet.			

3.3.2 Klasse C1

Stoffgebiet	Abschnitt im Fragenkatalog	Zahl der Fragen	Summe der Punkte
1. Grundstoff	(wie unter 3.2.2)	10	39
2. Zusatzstoff	2.1 bis 2.8	20	66
Gesamtstoff		30	105
Zulässige Fehlerpunkte 10; es sei denn, zwei Fragen mit Wertigkeit 5 falsch beantwortet.			

3.4 Zusammenstellung und Bewertung der Fragen für die Fahrerlaubnisklassen D und D1

3.4.1 Klasse D

Stoffgebiet	Abschnitt im Fragenkatalog	Zahl der Fragen	Summe der Punkte
1. Grundstoff	(wie unter 3.2.2)	10	39
2. Zusatzstoff	2.1 bis 2.8	30	99
Gesamtstoff		40	138
Zulässige Fehlerpunkte 10; es sei denn, zwei Fragen mit Wertigkeit 5 falsch beantwortet.			

3.4.2 Klasse D 1

Stoffgebiet	Abschnitt im Fragenkatalog	Zahl der Fragen	Summe der Punkte
1. Grundstoff	(wie unter 3.2.2)	10	39
2. Zusatzstoff	2.1 bis 2.8	25	82
Gesamtstoff		35	121
Zulässige Fehlerpunkte 10; es sei denn, zwei Fragen mit Wertigkeit 5 falsch beantwortet.			

3.5 Zusammenstellung und Bewertung der Fragen für die Fahrerlaubnisklasse CE

Stoffgebiet	Abschnitt im Fragenkatalog	Zahl der Fragen	Summe der Punkte
1. Grundstoff	(wie unter 3.2.2)	10	39
2. Zusatzstoff	2.1 bis 2.8	20	66
Gesamtstoff		30	105
Zulässige Fehlerpunkte 10; es sei denn, zwei Fragen mit Wertigkeit 5 falsch beantwortet.			

3.6 Zusammenstellung und Bewertung der Fragen für Bewerber um eine Mofa-Prüfbescheinigung

Stoffgebiet	Abschnitt im Fragenkatalog	Zahl der Fragen	Summe der Punkte
1. Grundstoff			
Gefahrenlehre	1.1	3 (davon 2 Bildfragen)	11
Verhalten im Straßenverkehr	1.2	4 (davon 2 Bildfragen)	13
Vorfahrt/Vorrang	1.3	3 (mindestens 2 Bildfragen)	15
Umweltschutz	1.5	1	3
Verkehrszeichen	1.4	4	12
Technik	1.8		
Eignung und Befähigung von Kraftfahrern	1.9		
Summe Grundstoff		15	54
2. Zusatzstoff	2.1 bis 2.8	5	15
Gesamtstoff		20	69
Zulässige Fehlerpunkte 7.			

3.7 Bewertung der Prüfung

3.7.1 Zulässige Fehlerpunkte einer einzelnen Klasse

	Zulässige Fehlerpunkte	
	Ersterwerb	Erweiterung ¹⁾
Klassen	20 Fragen Grundstoff plus Zusatzstoff der jeweiligen Klasse	10 Fragen Grundstoff
A, A1, B, M, S, L und T	10	6
C1	-	10
C	-	10
CE	-	10
D1	-	10
D	-	10

Die Prüfung ist nicht bestanden, wenn:

- 2 Fragen mit Wertigkeit 5 falsch beantwortet werden oder
- die zulässigen Fehlerpunkte überschritten werden.

Grund- und Zusatzstoff werden **immer** gemeinsam bewertet.

¹⁾ Um eine Erweiterungsprüfung handelt es sich immer dann, wenn eine bestandene, noch gültige theoretische Fahrerlaubnisprüfung oder eine Fahrerlaubnis vorhanden ist.

3.7.2 Zulässige Fehlerpunkte bei gleichzeitiger Prüfung mehrerer Klassen in einem Termin

	Zulässige Fehlerpunkte											
	Ersterwerb						Erweiterung ¹⁾					
Klassen	20 Fragen Grundstoff plus Zusatzstoff der jeweiligen Klassen						10 Fragen Grundstoff					
A, A1, B, M, S, L und T	10						6					
Zulässige Fehlerpunkte bei Klassen, die Klasse B voraussetzen												
	B	C1	C	CE	D1	D	B	C1	C	CE	D1	D
B + C1	10	13					6	10				
B + C	10		13				6		10			
B + C + CE	10		13	13			6		10	10		
B + D1	10				13		6				10	
B + D	10					13	6					10

Die Prüfung ist nicht bestanden, wenn:

- 2 Fragen mit Wertigkeit 5 falsch beantwortet werden oder
- die zulässigen Fehlerpunkte überschritten werden

Die zulässigen Fehlerpunkte weiterer Kombinationen von Fahrerlaubnisklassen in einem Prüfungstermin sind den Rubriken „Ersterwerb“ oder „Erweiterungen“ zu entnehmen.

Bei gleichzeitiger Prüfung mehrerer Klassen in einem Termin wird der Grundstoff nur einmal geprüft.

Grund- und Zusatzstoff werden **immer** gemeinsam bewertet.

¹⁾ Um eine Erweiterungsprüfung handelt es sich immer dann, wenn eine bestandene, noch gültige theoretische Fahrerlaubnisprüfung oder eine Fahrerlaubnis vorhanden ist.

3.8 Beispiele

Beispiel 1: Prüfung Klasse B (Ersterwerb)

Auswertung der Prüfung

Grundstoff }
Zusatzstoff } 10 Fehlerpunkte

Zwei Fragen mit Wertigkeit 5 falsch beantwortet.

Prüfungsergebnis:

Klasse B: Nicht bestanden, weil zwei Fragen mit Wertigkeit 5 falsch beantwortet (s. Nr. 3.2.1 und 3.7.1)

Beispiel 2: Prüfung Klasse A (Erweiterung)

Auswertung der Prüfung

Grundstoff: 4 Fehlerpunkte

Zusatzstoff Klasse A: 3 Fehlerpunkte

Prüfungsergebnis:

Klasse A: Nicht bestanden, weil mehr als 6 Fehlerpunkte (s. Nr. 3.2.2 und 3.7.1).

Beispiel 3: Gleichzeitige Prüfung der Klassen A und B (Ersterwerb)

Auswertung der Prüfung

Grundstoff: 5 Fehlerpunkte
 Eine Frage mit Wertigkeit 5 falsch beantwortet

Zusatzstoff Klasse A: 5 Fehlerpunkte
 Eine Frage mit Wertigkeit 5 falsch beantwortet

Zusatzstoff Klasse B: 5 Fehlerpunkte
 Keine Frage mit Wertigkeit 5 falsch beantwortet

Prüfungsergebnis

Klasse A: Nicht bestanden, weil zwei Fragen mit Wertigkeit 5 falsch beantwortet (s. Nr. 3.2.1 und 3.7.1)

Klasse B: Bestanden, (zusammen 10 (5+5) Fehlerpunkte, s. Nr. 3.2.1 und 3.7.1)

Beispiel 4: Gleichzeitige Prüfung Klasse B (Ersterwerb) und C und CE in einem Termin

Die theoretische Prüfung erfolgt schrittweise (Grundstoff der Klasse B wird bei allen Klassen berücksichtigt).

1. Schritt: Prüfung Klasse B

(Grundstoff für alle Klassen (Ersterwerb) und Zusatzstoff für Klasse B)

Grundstoff: 5 Fehlerpunkte

Eine Frage mit Wertigkeit 5 falsch beantwortet

Zusatzstoff: 4 Fehlerpunkte

Prüfungsergebnis:

Klasse B: Bestanden (s. Nr. 3.2.1 und 3.7.1),
Fortsetzung mit Prüfung Klasse C (2. Schritt)

2. Schritt: Prüfung Klasse C

Auswertung des Zusatzstoffes Klasse C

Grundstoff (s. 1. Schritt): 5 Fehlerpunkte

Eine Frage mit Wertigkeit 5 falsch beantwortet

Zusatzstoff Klasse C: 8 Fehlerpunkte

Prüfungsergebnis:

Klasse C: Bestanden (s. Nr. 3.7.2, Ersterwerb)
Fortsetzung mit Prüfung Klasse CE (3. Schritt)

3. Schritt: Prüfung Klasse CE

Auswertung des Zusatzstoffes Klasse CE

Grundstoff (s. 1. Schritt): 5 Fehlerpunkte

Eine Frage mit Wertigkeit 5 falsch beantwortet

Zusatzstoff Klasse CE: 8 Fehlerpunkte

Prüfungsergebnis:

Klasse CE: Bestanden (s. Nr. 3.7.2, Ersterwerb)

Hinweis:

Bei Nichtbestehen der theoretischen Prüfung Klasse C kann auf Wunsch des Bewerbers die praktische Prüfung B abgelegt werden. Analog kann bei Nichtbestehen der Klasse CE die praktische Prüfung in Klasse B und C abgelegt werden.

Beispiel 5: Prüfung der Klasse D**Auswertung der Prüfung**

Grundstoff: 6 Fehlerpunkte

Zusatzstoff Klasse D: 5 Fehlerpunkte

Prüfungsergebnis

Klasse D: Nicht bestanden, weil mehr als 10 Fehlerpunkte (6+5), (s. Nr. 3.4.1 und 3.7.1)

Wahrscheinlichkeitsrechnung Lottozahlen

Mit Hilfe der Wahrscheinlichkeitsrechnung lässt sich die Anzahl der möglichen Zahlenkombinationen beim Lotto errechnen. Beim Spiel 6 aus 49 kann man aus 49 Zahlen 6 ohne Beachtung der Reihenfolge auswählen.

$$\frac{49 \cdot 48 \cdot 47 \cdot 46 \cdot 45 \cdot 44}{6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1} = \frac{10.068.347.520}{720} = 13.983.816$$

Dabei gibt es in der Summe genau 13.983.816 Möglichkeiten. Wenn man nun eine Reihe auf dem Lottoschein ausfüllt, hat man also etwa eine Chance von 1 zu 14 Millionen, einen sechser im Lotto zu erhalten. Um das Verhältnis zu verbessern kann man nur mehrere Reihen, bzw. Scheine ausfüllen. Bei 20 Reihen stehen die Chancen demnach 1 zu 700000. Wenn man nun aber die Superzahl mit einbezieht, wächst diese Zahl wieder auf das zehnfache, also 7 Millionen, bzw. 140 Millionen bei einer Reihe.

All you need is luck

luck luck luck luck luck luck
luck luck luck luck luck luck

there's nothing you can win, the curse is on,
every year this time your heads are hung
nothing you can do, it's just not your destiny
believe me..

nothing you can do no luck abounds
not even tossing Zimmer to the ground
no one there can play,
see Buckner boot the ball away,
believe me

all you need is luck
all you need is luck
all you ned is luck, luck
good luck is all you need

the pennant race is over, you have shown,
every little lead you have gets blown
all you've got to show, is that curse that will tease your game
it's easy

all you need is luck
all you need is luck
all you ned is luck, luck
good luck is all you need

6.5 Die Grenzen der Programme

Jedes Programm kommt irgendwann einmal an seine Grenzen, so auch unsere Programme. Um die Eingabe möglichst großer Werte für „k“ und „n“ im Führerschein-Programm (und anderen Parametern in den anderen Programmen) zu ermöglichen, wurden die Zahlenbereiche für Gleitkommazahlen auf den Datentyp „Extended“ erweitert. Der Zahlenbereich für „Extended“ geht von $3,6 \times 10^{-4951}$ bis $1,1 \times 10^{4932}$.

Zum Vergleich: Der übliche Typ für Gleitkommazahlen „Real“ hat nur einen Datenbereich von $5,0 \times 10^{-324}$ bis $1,7 \times 10^{308}$. Während man im Bereich von „Extended“ noch die Fakultät von 1700 ausrechnen kann, schafft der „Real“-Typ nur die Fakultät bis ca. 170.

6.6 Umsetzung des Newton-Verfahrens

Nach dem Newton-Verfahren wird die Nullstelle einer Funktion folgendermaßen angenähert:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

In diesem Programm wird das Newton-Verfahren durch eine selbstgeschriebene Funktion realisiert:

```
function TForm1.newton(x:Extended):Extended;
var i:Integer;
begin
  i:=0;
  repeat
    x:=x-(abl(x)/abl2(x));
    if x >= maxdef then x:=maxdef;
    if debug then Form1.Caption:=FloatToStr(abs(abl(x))) + ' --- ' +
IntToStr(i) + ' --- ' + FloatToStr(x);
    i:=i+1;
  until (abs(abl(x)) < (1/1000000)) or (i = 1000);
  result:=x;
  if debug then EdDebug.Text:=EdDebug.Text + 'x=' + FloatToStr(result)+ ' ';
';
end;
```

Bei i handelt es sich um eine Zählvariable für die repeat-Schleife, die nach jedem Schleifendurchlauf um 1 erhöht wird. Nachdem i auf 0 gesetzt wurde, wird in der Schleife zunächst der neue Wert für x ausgerechnet. Dabei berechnet die Funktion abl(x) das $f(x_n)$ und die Funktion abl2(x) das $f'(x_n)$. Der erste Wert für x_n wird beim Funktionsaufruf mittels der Variable x übergeben. Denn streng genommen handelt es sich bei dem $f(x_n)$ um den Teil von $p'(x)$, der gleich Null werden kann, und $f'(x_n)$ wäre eine Art zweite Ableitung – daher auch die Bezeichnung „abl“ und „abl2“ für Ableitung.

Anschließend wird überprüft, ob das neue x außerhalb des Definitionsbereiches liegt. Wenn dies der Fall ist, wird x auf den größtmöglichen Wert gesetzt, der zuvor ausgerechnet wurde und in der Variablen „maxdef“ gespeichert ist.

Nachdem ggf. noch das ungerundete Zwischenergebnis ausgegeben wird, falls der Debug-Modus aktiviert ist, wird die Zählvariable i um 1 erhöht.

Die Rechnung wird in der repeat-Schleife so lange wiederholt, bis $f(x)$ kleiner als ein millionstel ist, oder wenn die Schleife 1000 Mal durchlaufen wurde.

6.7 Berechnung der Fakultät

```
function TForm1.Fakultaet(zahl: Integer):Extended;
var i: Integer;
    fak: Extended;
```

```

begin
    fak:=1;
    for i:=1 to zahl do
        fak:=fak*i;
    result:=fak;
end;

```

Bei der Variablen i handelt es sich um eine Zählvariable für eine For-Schleife, die Zwischenergebnisse der Fakultät werden in der Variablen „fak“ gespeichert. Beim Initialisieren wird „fak“ auf 1 gesetzt, anschließend das Zwischenergebnis in einer Schleife mit dem aktuellen Wert von i multipliziert.

Die Schleife wird solange durchlaufen, bis i gleich der Zahl ist, von der die Fakultät berechnet werden soll. Am Ende steht die Fakultät in der Variablen „fak“ und wird mittels „result:=fak“ als Rückgabewert der Funktion definiert.

6.8 Überprüfung auf eine gültige Gleitkommazahl

Damit die Programme funktionieren, muss u.a. sichergestellt werden, dass dort, wo Zahlenwerte einzutragen sind, auch wirklich Zahlenwerte und keine Buchstaben etc. eingetragen werden. Da Delphi keine Funktion zur Überprüfung von Zahlenwerten bereitstellt, haben wir eine eigene Funktion IsFloat geschrieben. Diese Funktion überprüft, ob der Inhalt eines Textfeldes auch wirklich eine gültige Gleitkommazahl ist. Der Rückgabewert ist entweder „true“ (Textfeld enthält eine gültige Gleitkommazahl) oder „false“ (Textfeld enthält keine gültige Gleitkommazahl):

```

function TForm1.IsFloat(Feld: TEdit):Boolean;
var i:Integer;
    komma: Integer;
    e: Integer;
    exponent: String;
    minus: Integer;
    pos: Integer;
begin
    result:=true;
    komma:=0;
    e:=0;
    minus:=0;
    pos:=0;
    for i:=1 to Length(Feld.Text) do
        case Feld.Text[i] of
            '0': result:=true;
            '1': result:=true;
            '2': result:=true;
            '3': result:=true;
            '4': result:=true;
            '5': result:=true;
            '6': result:=true;
            '7': result:=true;
            '8': result:=true;
            '9': result:=true;
            ',': begin result:=true; komma:=komma+1; pos:=i; end;
            else
                begin
                    result:=false;
                    exit;
                end;
            end;
    end;

    if (komma > 1) or (e > 1) or (minus > 1) then
        begin
            result:=false;
            exit;
        end;
end;

```

```

end;

if Feld.Text[1] = ',' then
begin
    result:=false;
    exit;
end;
if (Feld.Text = '') or (pos > 360) then result:=false;
end;

```

Die Variable `i` stellt eine Zählvariable dar, in den Variablen „komma“ bzw. „e“ wird die gefundene Anzahl an Kommata bzw. „E“ (das E steht für die Zehnerpotenzen, $2,67E10$ entspricht $2,67 * 10^{10}$) in dem zu überprüfenden Textfeld gespeichert. Das zu überprüfende Textfeld wird beim Funktionsaufruf mit der Variablen „Feld“ übermittelt. In der Variablen „exponent“ wird der Wert des Exponenten der Zehnerpotenz gespeichert – im vorigen Beispiel hätte „exponent“ den Wert „10“. Die Variable wurde absichtlich als String deklariert, da eine 10-stellige Zahl bereits den Integer-Bereich überschreiten könnte. Daher wird später in der Funktion erst die Länge der Zeichenkette überprüft, und falls diese kleiner/gleich als vier ist (der kleinstmögliche Exponent wäre -4950, der größte +4931, ansonsten wird auch der Extended-Bereich überschritten) in einen Integer umgewandelt und geprüft, ob diese kleiner/gleich 4950 ist.

Die Variable „minus“ speichert die Anzahl der gefundenen Minus-Zeichen, und „pos“ speichert die Position der ersten Ziffer des Exponenten, falls ein Exponent vorhanden ist. Nach dem Initialisieren der Variablen wird jedes einzelne Zeichen des Textfeldes mit der Case-Anweisung geprüft. Falls das Zeichen weder eine Ziffer, noch ein „-“, „E“, „e“ (für Zehnerpotenzen) oder „-“, ist, wird der Rückgabewert der Funktion auf „false“ gesetzt. Falls das Zeichen ein „E“, „e“ oder „-“, ist, merkt sich das Programm die Position des Zeichens, um den Exponenten aus der Zeichenkette zu „schneiden“. Nach dem Durchlaufen der Case-Anweisung ist in der Variablen „pos“ die Position des ersten Zeichens des Exponenten gespeichert.

Dann folgen verschiedene Überprüfungen, bei denen der Rückgabewert auf „false“ gesetzt und die Funktion abgebrochen wird, falls das Ergebnis der Überprüfung negativ ist:

Als erstes wird überprüft, ob der Benutzer mehrere Minuszeichen, Kommata oder „e“, „E“ eingegeben hat. Danach wird die Stelle des Minus-Zeichens überprüft, da das Minus nur unmittelbar nach dem „e“, „E“ stehen darf. Denn „12,48E8-25“ ist keine gültige Gleitkommazahl.

Nun wird „schneidet“ das Programm den Exponenten (falls vorhanden) heraus. Da sich das Programm die Stelle der ersten Ziffer des Exponenten gemerkt hat, ist der Exponent der Teil der Zeichenkette von der ersten Ziffer des Exponenten bis zum Ende der Zeichenkette.

Anschließend wird noch geprüft, ob der Exponent die oben genannten Bedingungen erfüllt (kleiner/gleich -4950 oder +4931), um nicht den Datenbereich von Extended-Variablen zu überschreiten.

Am Ende der Funktion kommt noch eine Überprüfung der Position der Kommata, „e“, „E“ und Minus-Zeichen, da diese nicht am Anfang der Zeichenkette stehen dürfen.

6.9 Überprüfung auf eine gültige ganze Zahl

Die Überprüfung auf eine gültige Integer-Zahl verläuft analog zur Überprüfung auf Gleitkommazahlen und wird daher nicht näher erläutert:

```

function TForm1.IsInteger(Feld: TEdit):Boolean;
var i:Integer;
begin
    result:=true;
    for i:=1 to Length(Feld.Text) do
        case Feld.Text[i] of

```

```

        '0': result:=true;
        '1': result:=true;
        '2': result:=true;
        '3': result:=true;
        '4': result:=true;
        '5': result:=true;
        '6': result:=true;
        '7': result:=true;
        '8': result:=true;
        '9': result:=true;
    else
    begin
        result:=false;
        exit;
    end;
end;
if (Feld.Text = '') or (Length(Feld.Text) >9) then result:=false;
end;

```

Berechnung der Gesamtwahrscheinlichkeit beim Führerschein-Programm

Beim Führerschein-Programm wird die Gesamtwahrscheinlichkeit der ausgewählten Bereiche automatisch berechnet. Die dazu geschriebene Prozedur benötigt eine Feld-Variable „Array“, die beim Programmstart gefüllt wird:

//Gefahrenlehre

```

db[1,1]:=8;
db[1,2]:=30;
db[1,3]:=0;
db[1,4]:=11/100;
db[1,5]:=89/100;

```

//Verhalten im Str-Verkehr

```

db[2,1]:=6;
db[2,2]:=30;
db[2,3]:=0;
db[2,4]:=10/100;
db[2,5]:=90/100;

```

//Vorfahrt

```

db[3,1]:=3;
db[3,2]:=30;
db[3,3]:=0;
db[3,4]:=13/43;
db[3,5]:=30/43;

```

//Verkehrszeichen

```

db[4,1]:=2;
db[4,2]:=30;
db[4,3]:=0;
db[4,4]:=6/100;
db[4,5]:=94/100;

```

//Umweltschutz

```

db[5,1]:=1;
db[5,2]:=30;
db[5,3]:=0;

```



```
db[5,4]:=4/44;
db[5,5]:=40/44;
```

```
//Zusatzfragen
db[6,1]:=10;
db[6,2]:=30;
db[6,3]:=4/100;
db[6,4]:=17/100;
db[6,5]:=79/100;
```

Diese Feldvariable enthält also die Daten folgender Tabelle:

Themengebiet	Anteil am Bogen	0 Antworten	2 Antworten	3 Antworten
Gefahrenlehre	8/30	0	11/100	89/100
Verhalten im Straßenverkehr	6/30	0	10/100	90/100
Vorfahrt	3/30	0	13/43	30/43
Verkehrszeichen	2/30	0	6/100	94/100
Umweltschutz	1/30	0	4/44	40/44
Zusatzstoff	10/30	4/100	17/100	79/100

Um aus den Daten auch die Anzahl der Fragen automatisch berechnen zu können, wurde die zweite Spalte „Anteil am Bogen“ jeweils in Zähler und Nenner aufgeteilt.

Nun die Prozedur:

```
procedure TForm1.calcbereiche(var bereich: Extended; bnr: Integer; cbox:
TCheckBox);
begin
  if cbox.Checked = true then
    begin
      anzfragen:=anzfragen + db[bnr,1];
      bereich:=(1/3)*(db[bnr,4]) + (1/7)*(db[bnr,5]);
    end
  else
    begin
      bereich:=0;
      anzfragen:=anzfragen - db[bnr,1];
    end;
  if (abs(wbereiche) < 0.00001) or (round(anzfragen) = 0) then
    begin
      Edpz.Text:='1';
      Edpn.Text:='';
      Edpz.SetFocus;
      Edn.Text:='';
    end;
end;
```

Beim Funktionsaufruf (für die Gefahrenlehre: `calcbereiche(wb2, 2, CheckBox2);`) werden noch folgende Parameter übergeben:

- Der Bereich, der aus- bzw. ausgewählt wurde (als Variable!)
- Die Nummer des Bereiches (Gefahrenlehre = 1, Verhalten im Straßenverkehr = 2 usw.)
- Der Name der Checkbox

Zunächst wird überprüft, ob der Bereich aus- oder ausgewählt wurde. Ist die Checkbox aktiviert, so wurde der Bereich ausgewählt, ansonsten wurde er abgewählt.

Als erstes wird in beiden Fällen die Anzahl der Fragen neu berechnet, indem die Anzahl der Fragen des Bereiches aus der Feldvariablen geholt und addiert bzw. subtrahiert werden.

Falls der Bereich ausgewählt wurde, wird die Wahrscheinlichkeit des Bereiches ausgerechnet und der Wert der beim Funktionsaufruf übergebenen Variablen zugeordnet. Wurde der Bereich abgewählt, wird die Wahrscheinlichkeit auf Null gesetzt.

Das Eintragen der Gesamtwahrscheinlichkeit in ein Textfeld übernimmt folgende Programmzeile eines Timers:

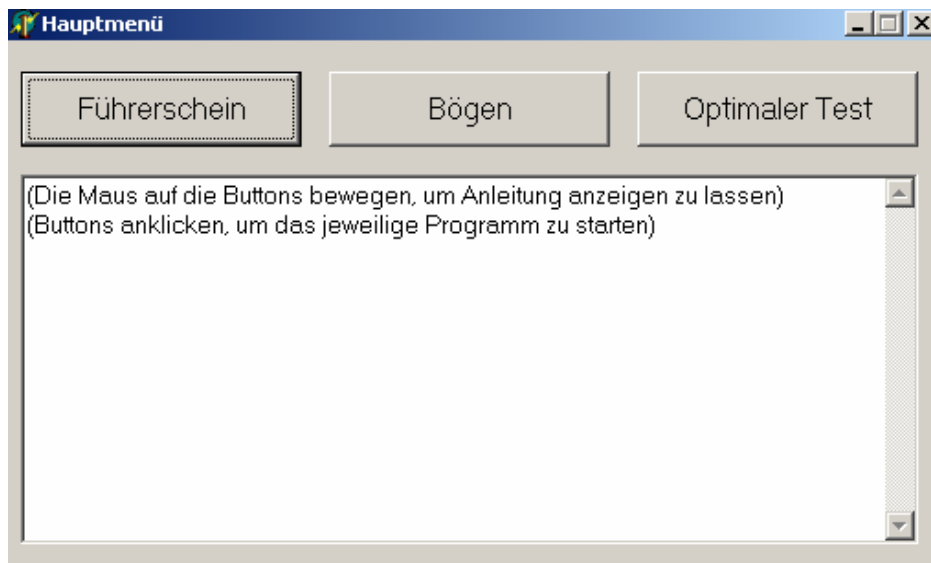
```
wbereiche:=(db[1,1])/(anzfragen)*wb1+(db[2,1])/(anzfragen)*wb2+(db[3,1])/(a
nzfragen)*wb3+(db[4,1])/(anzfragen)*wb4+(db[5,1])/(anzfragen)*wb5+(db[6,1])
/(anzfragen)*wb6;
```

Es wird dann eine Summe gebildet, die aus dem Anteil an den Gesamtfragen der jeweiligen Bereiche multipliziert mit der Wahrscheinlichkeit besteht. Ist der Bereich nicht ausgewählt, beträgt die Wahrscheinlichkeit für den Bereich Null und folglich ist der Summand ebenfalls Null.

Sind alle Bereiche abgewählt, so beträgt die Gesamtwahrscheinlichkeit aufgrund von Ungenauigkeiten bei Gleitkommazahlen nur ungefähr Null und nicht exakt Null. Wenn die Wahrscheinlichkeit kleiner als 0,00001 (also fast Null) beträgt, geht das Programm davon aus, dass keine Bereiche ausgewählt sind und setzt die Eingabefelder in einen Ausgangszustand zurück.

6.10 Quelltexte

Hauptmenue.exe:



Screenshot: Hauptmenü, um die anderen Programme zu starten

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms,
  Dialogs, StdCtrls, ShellApi;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
```

```

    Memol: TMemo;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button1MouseMove(Sender: TObject; Shift: TShiftState; X,
        Y: Integer);
    procedure Button2MouseMove(Sender: TObject; Shift: TShiftState; X,
        Y: Integer);
    procedure Button3MouseMove(Sender: TObject; Shift: TShiftState; X,
        Y: Integer);
private
    { Private declarations }
public
    { Public declarations }
    procedure start(anwendung:string);
end;

var
    Form1: TForm1;
    prog:Integer;

implementation

{$R *.dfm}
//#####Eigene Prozeduren

procedure TForm1.start(anwendung:String);
var i:Integer;
begin
    i:=ShellExecute(handle, 'open', PChar(anwendung), PChar(''), NIL,
SW_SHOW);
    if i <= 32 then showmessage('Fehler: Das gewünschte Programm konnte nicht
gestartet werden.');
```

```

end;

//###Programm Prozeduren

procedure TForm1.Button1Click(Sender: TObject);
begin
    start('fuehrerschein.exe');
```

```

end;

procedure TForm1.Button2Click(Sender: TObject);
begin
    start('boegen.exe');
```

```

end;

procedure TForm1.Button3Click(Sender: TObject);
begin
    start('optimaler_ankreuzbogen.exe');
```

```

end;

procedure TForm1.Button1MouseMove(Sender: TObject; Shift: TShiftState; X,
    Y: Integer);
begin
    if prog <> 1 then Memol.Lines.LoadFromFile('fuehrerschein.txt');
```

```

    prog:=1;
end;

procedure TForm1.Button2MouseMove(Sender: TObject; Shift: TShiftState; X,
    Y: Integer);
begin
    if prog <> 2 then Memol.Lines.LoadFromFile('boegen.txt');
```

```

    prog:=2;
end;

procedure TForm1.Button3MouseMove(Sender: TObject; Shift: TShiftState; X,
```

```

    Y: Integer);
begin
    if prog <> 3 then Memo1.Lines.LoadFromFile('optimaler_test.txt');
    prog:=3;
end;

end.

```

Führerschein.exe:

Screenshot: Die Bereiche Gefahrenlehre, Umweltschutz und Zusatzstoff sollen durch Raten bestanden werden. Dazu müssen im Durchschnitt 17 von 19 Fragen richtig beantwortet werden, die Wahrscheinlichkeit dafür beträgt $7,14 \cdot 10^{-12}$

```

unit Unit1;

interface

uses
    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
    Forms,
    Dialogs, ExtCtrls, StdCtrls, Math, Buttons;

type
    TForm1 = class(TForm)
        Label1: TLabel;
        Label2: TLabel;
        Label3: TLabel;
        EdPz: TEdit;
        EdPn: TEdit;
        Shapel: TShape;
        Edk: TEdit;
        Edn: TEdit;
        BtCalc: TButton;
        Label4: TLabel;
        Label5: TLabel;
        Label6: TLabel;
        GroupBox1: TGroupBox;
        EdK1: TEdit;
        EdK2: TEdit;
        EdK3: TEdit;
        Edk4: TEdit;
        EdK5: TEdit;
        EdK6: TEdit;
        GroupBox2: TGroupBox;
        EdErgebnis: TEdit;
        Button1: TButton;
    end;

```

```

Button2: TButton;
GroupBox3: TGroupBox;
Label7: TLabel;
CheckBox1: TCheckBox;
CheckBox2: TCheckBox;
CheckBox3: TCheckBox;
CheckBox4: TCheckBox;
CheckBox5: TCheckBox;
CheckBox6: TCheckBox;
Timer1: TTimer;
CheckBox7: TCheckBox;
Shape2: TShape;
CheckBox8: TCheckBox;
Label8: TLabel;
Memo1: TMemo;
Memo2: TMemo;
procedure BtCalcClick(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure Timer1Timer(Sender: TObject);
procedure CheckBox1Click(Sender: TObject);
procedure CheckBox2Click(Sender: TObject);
procedure CheckBox3Click(Sender: TObject);
procedure CheckBox4Click(Sender: TObject);
procedure CheckBox6Click(Sender: TObject);
procedure CheckBox5Click(Sender: TObject);
procedure CheckBox7Click(Sender: TObject);
procedure CheckBox8Click(Sender: TObject);
private
{ Private declarations }
function Fakultaet(zahl: Integer):Extended;
function IsInteger(Feld: TEdit):Boolean;
function IsFloat(Feld: TEdit):Boolean;
function CheckEingabe:Boolean;
procedure showbereiche;
procedure showhinweise;
procedure calcbereiche(var bereich: Extended; bnr: Integer; cbox:
TCheckBox);
public
{ Public declarations }
end;

var

//Variable k entspricht der Eingabe i !!!
Form1: TForm1;
pz: Extended;
pn: Int64;
k: Int64;
n: Int64;
matrix: Extended;
matrixz: Extended;
matrixn: Extended;
faktor1: Extended;
faktor2: Extended;
exponent: Extended;
b: Extended;
ergebnis: Extended;
wbereiche: Extended;
anzfragen: Extended;
db: Array[1..6,1..5] of real;
wb1: Extended;
wb2: Extended;
wb3: Extended;
wb4: Extended;
wb5: Extended;

```

```

wb6: Extended;

implementation

{$R *.dfm}

//Eigene Funktionen und Prozeduren
function TForm1.Fakultaet(zahl: Integer):Extended;
var i: Integer;
    fak: Extended;
begin
    fak:=1;
    for i:=1 to zahl do
        fak:=fak*i;
    result:=fak;
end;

function TForm1.IsInteger(Feld: TEdit):Boolean;
var i:Integer;
begin
    result:=true;
    for i:=1 to Length(Feld.Text) do
        case Feld.Text[i] of
            '0': result:=true;
            '1': result:=true;
            '2': result:=true;
            '3': result:=true;
            '4': result:=true;
            '5': result:=true;
            '6': result:=true;
            '7': result:=true;
            '8': result:=true;
            '9': result:=true;
        else
            begin
                result:=false;
                exit;
            end;
        end;
    end;
    if (Feld.Text = '') or (Length(Feld.Text) >9) then result:=false;
end;

function TForm1.IsFloat(Feld: TEdit):Boolean;
var i:Integer;
    komma: Integer;
    e: Integer;
    exponent: String;
    minus: Integer;
    pos: Integer;
begin
    result:=true;
    komma:=0;
    e:=0;
    minus:=0;
    for i:=1 to Length(Feld.Text) do
        case Feld.Text[i] of
            '0': result:=true;
            '1': result:=true;
            '2': result:=true;
            '3': result:=true;
            '4': result:=true;
            '5': result:=true;
            '6': result:=true;
            '7': result:=true;
            '8': result:=true;
            '9': result:=true;
            ',': begin result:=true; komma:=komma+1; end;

```

```

        'E': begin result:=true; e:=e+1; pos:=i+1 end;
        'e': begin result:=true; e:=e+1; pos:=i+1 end;
        '-': begin result:=true; minus:=minus+1; pos:=i+1; end;
    else
    begin
        result:=false;
        exit;
    end;
end;

if (komma > 1) or (e > 1) or (minus > 1) then
begin
    result:=false;
    exit;
end;

if (e = 1) and (minus = 1) then
    if (Feld.Text[pos-2] <> 'e') and (Feld.Text[pos-2] <> 'E') then
    begin
        result:=false;
        exit;
    end;

if e = 1 then
begin
    showmessage(IntToStr(length(Feld.Text)));
    exponent:=Copy(Feld.Text, pos, length(Feld.Text)-pos+1);
    if length(exponent) <= 4 then
        if (minus = 1) and (StrToInt(exponent) > 4950) then
        begin
            result:=false;
            exit;
        end
        else if (minus = 0) and (StrToInt(exponent) > 4931) then
        begin
            result:=false;
            exit;
        end;
        if length(exponent) > 4 then
        begin
            result:=false;
            exit;
        end;
    end;

    if (Feld.Text[1] = ',') or (Feld.Text[1] = 'E') or (Feld.Text[1] = '-')
then
    begin
        result:=false;
        exit;
    end;
    if (Feld.Text = '') or (Length(Feld.Text) > 50) then result:=false;
end;

function TForm1.CheckEingabe:Boolean;
begin
    result:=true;

    if not IsFloat(Edpz) then
    begin
        showmessage('Fehler: Der Zähler von p ist keine gültige Zahl zwischen 1
und 500.');
```

```

else if (IsFloat(Edpz)) then
    if StrToFloat(Edpz.Text) > 500 then
        begin
            showmessage('Fehler: Der Zähler von p ist keine gültige Zahl zwischen
1 und 500.');
```

```

            result:=false;
            exit;
        end;

    if not IsInteger(Edpn) then
        begin
            showmessage('Fehler: Der Nenner von p ist keine gültige natürliche Zahl
zwischen 1 und 500.');
```

```

            result:=false;
            exit;
        end
    else if (IsInteger(Edpn)) then
        if StrToInt(Edpn.Text) > 500 then
            begin
                showmessage('Fehler: Der Nenner von p ist keine gültige natürliche
Zahl zwischen 1 und 500.');
```

```

                result:=false;
                exit;
            end;

        if (IsInteger(Edpz)) and (IsInteger(Edpn)) then
            if (StrToFloat(Edpz.Text) / StrToFloat(Edpn.Text) < 0) or
(StrToFloat(Edpz.Text) / StrToFloat(Edpn.Text) > 1) then
                begin
                    showmessage('Fehler: Die Wahrscheinlichkeit p liegt nicht zwischen 0
und 1.');
```

```

                    result:=false;
                    exit;
                end;

            if not IsInteger(Edk) then
                begin
                    showmessage('Fehler: i ist keine gültige natürliche Zahl zwischen 1 und
1700.');
```

```

                    result:=false;
                    exit;
                end
            else if (IsInteger(Edk)) then
                if StrToInt(Edk.Text) > 1700 then
                    begin
                        showmessage('Fehler: i ist keine gültige natürliche Zahl zwischen 1
und 1700.');
```

```

                        result:=false;
                        exit;
                    end;

                if not IsInteger(Edn) then
                    begin
                        showmessage('Fehler: n ist keine gültige natürliche Zahl zwischen 1 und
1700.');
```

```

                        result:=false;
                        exit;
                    end
                else if (IsInteger(Edn)) then
                    if StrToInt(Edn.Text) > 1700 then
                        begin
                            showmessage('Fehler: n ist keine gültige natürliche Zahl zwischen 1
und 1700.');
```

```

                            result:=false;
                            exit;
                        end;

```



```

    if (IsInteger(Edk)) and (IsInteger(Edn)) then
        if StrToInt(Edk.Text) > StrToInt(Edn.Text) then
            begin
                showmessage('Fehler: i > n');
                result:=false;
                exit;
            end;
        end;

end;

procedure TForm1.showbereiche;
begin
    //Zeigt die Bereiche der Theoretischen Führerscheinprüfung an
    Label4.Visible:=false;
    Label5.Visible:=false;
    Label6.Visible:=false;
    Label7.Visible:=false;
    Memo1.Visible:=false;
    Memo2.Visible:=false;
    GroupBox3.Visible:=true;
end;

procedure TForm1.showhinweise;
begin
    //Zeigt die Hinweise für die einzelnen Felder an
    Label4.Visible:=true;
    Label5.Visible:=true;
    Label6.Visible:=true;
    Label7.Visible:=true;
    Memo1.Visible:=true;
    Memo2.Visible:=true;
    GroupBox3.visible:=false;
end;

procedure TForm1.calcbereiche(var bereich: Extended; bnr: Integer; cbox:
TCheckBox);
begin
    if cbox.Checked = true then
        begin
            anzfragen:=anzfragen + db[bnr,1];
            bereich:=(1/3)*(db[bnr,4]) + (1/7)*(db[bnr,5]);
        end
        else
            begin
                bereich:=0;
                anzfragen:=anzfragen - db[bnr,1];
            end;
    if (abs(wbereiche) < 0.00001) or (round(anzfragen) = 0) then
        begin
            Edpz.Text:='1';
            Edpn.Text:='';
            Edpz.SetFocus;
            Edn.Text:='';
        end;
end;

//Programm

procedure TForm1.BtCalcClick(Sender: TObject);
var i:Integer;
begin
    //W = matrix * p^k * (1-p)^(n-k)
    // p^k = faktor1, (1-p) = faktor2, (n-k) = exponent
    //Eingaben prüfen
    if not CheckEingabe then exit;
    //Datein einlesen
    pz:=StrToFloat(edpz.text);

```

```

pn:=StrToInt(edpn.text);
k:=StrToInt(edk.text);
n:=StrToInt(edn.text);
//Variable Ergebnis initialisieren
ergebnis:=0;
//Ergebnisse berechnen
EdK6.Text:='';
for i:=k to n do
begin
    matrixz:=Fakultaet(n);
    EdK1.Text:='n! = ' + FloatToStr(Matrixz);
    EdK2.Text:='i! = ' + FloatToStr(Fakultaet(k)) + '; (n-i)! = ' +
FloatToStr(Fakultaet((n-k)));
    matrixn:=Fakultaet(i)*Fakultaet((n-i));
    EdK3.Text:='i!*(n-i)! = ' + FloatToStr(matrixn);
    matrix:=matrixz / matrixn;
    faktor1:=(Power(pz, i))/(Power(pn, i));
    EdK4.Text:='p^i = ' + FloatToStr((Power(pz, k))/(Power(pn, k)));
    faktor2:=((1 * pn - pz) / pn);
    EdK5.Text:='1-p = ' + FloatToStr(faktor2);
    Exponent:=n - i;
    B:=matrix * faktor1 * (Power(faktor2, Exponent));
    EdK6.Text:= EdK6.Text + 'P(' + IntToStr(i) + ') = ' + FloatToStrF(B,
ffixed, 10,5) + '; ';
    ergebnis:=ergebnis+B;
end;
//showmessage(FloatToStr(Ergebnis));
EdErgebnis.Text:=FloatToStr(Ergebnis);
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
    showhinweise;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
    showbereiche;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
    showbereiche;
    wbereiche:=0;
    //Array mit Daten der Bereiche füllen
    //Gefahrenlehre
    db[1,1]:=8;
    db[1,2]:=30;
    db[1,3]:=0;
    db[1,4]:=11/100;
    db[1,5]:=89/100;

    //Verhalten im Str-Verkehr
    db[2,1]:=6;
    db[2,2]:=30;
    db[2,3]:=0;
    db[2,4]:=10/100;
    db[2,5]:=90/100;

    //Vorfahrt
    db[3,1]:=3;
    db[3,2]:=30;
    db[3,3]:=0;
    db[3,4]:=13/43;
    db[3,5]:=30/43;

    //Verkehrszeichen

```

```

db[4,1]:=2;
db[4,2]:=30;
db[4,3]:=0;
db[4,4]:=6/100;
db[4,5]:=94/100;

//Umweltschutz
db[5,1]:=1;
db[5,2]:=30;
db[5,3]:=0;
db[5,4]:=4/44;
db[5,5]:=40/44;

//Zusatzfragen
db[6,1]:=10;
db[6,2]:=30;
db[6,3]:=4/100;
db[6,4]:=17/100;
db[6,5]:=79/100;

anzfragen:=0;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
    if (CheckBox1.Checked = true) or (CheckBox2.Checked = true) or
    (CheckBox3.Checked = true) or (CheckBox4.Checked = true) or
    (CheckBox5.Checked = true) or (CheckBox6.Checked = true) then
        begin
            wbereiche:=(db[1,1])/(anzfragen)*wb1+(db[2,1])/(anzfragen)*wb2+(db[3,1])/(a
            nzfragen)*wb3+(db[4,1])/(anzfragen)*wb4+(db[5,1])/(anzfragen)*wb5+(db[6,1])
            /(anzfragen)*wb6;
            Edpz.Text:=FloatToStr(wbereiche);
            CheckBox8.Enabled:=true;
            Edpn.Text:='1';
            if CheckBox8.Checked = true then
                Edn.Text:=FloatToStr(round(anzfragen));
            end
            else
                begin
                    CheckBox8.Checked:=false;
                    CheckBox8.Enabled:=false;
                end;
            if (round(anzfragen) >= 2) and (CheckBox8.Checked = true) then
                begin
                    CheckBox7.Enabled:=true;
                    if CheckBox7.Checked = true then EdK.Text:=FloatToStr(round(anzfragen -
                    2));
                end
                else
                    begin
                        CheckBox7.Checked:=false;
                        CheckBox7.Enabled:=false;
                    end;
            end;

        end;

end;

procedure TForm1.CheckBox1Click(Sender: TObject);
begin
    calcbereiche(wb1, 1, CheckBox1);
end;

procedure TForm1.CheckBox2Click(Sender: TObject);
begin
    calcbereiche(wb2, 2, CheckBox2);
end;

```

```

procedure TForm1.CheckBox3Click(Sender: TObject);
begin
    calcbereiche(wb3, 3, CheckBox3);
end;

procedure TForm1.CheckBox4Click(Sender: TObject);
begin
    calcbereiche(wb4, 4, CheckBox4);
end;

procedure TForm1.CheckBox6Click(Sender: TObject);
begin
    calcbereiche(wb6, 6, CheckBox6);
end;

procedure TForm1.CheckBox5Click(Sender: TObject);
begin
    calcbereiche(wb5, 5, CheckBox5);
end;

procedure TForm1.CheckBox7Click(Sender: TObject);
begin
    if CheckBox7.Checked = true then EdK.Text:=IntToStr(28)
    else EdK.Text:='';
end;

procedure TForm1.CheckBox8Click(Sender: TObject);
begin
    if CheckBox8.Checked = true then Edn.Text:=IntToStr(round(anzfragen))
    else Edn.Text:='';
end;

end.

```

Boegen.exe:

Screenshot: Die Wahrscheinlichkeit, die gesamte Testserie zu bestehen, beträgt ca. 0,04

```

unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms,
  Dialogs, ExtCtrls, StdCtrls, Math;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    EdPz: TEdit;
    EdPn: TEdit;
    Shape1: TShape;
    Edk: TEdit;
    Edn: TEdit;
    BtCalc: TButton;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    GroupBox1: TGroupBox;
    EdK1: TEdit;
    EdK2: TEdit;
    EdK3: TEdit;
    Edk4: TEdit;
    EdK5: TEdit;
    EdK6: TEdit;
    GroupBox2: TGroupBox;
    EdErgebnis: TEdit;
    Button1: TButton;
    Button2: TButton;
    GroupBox3: TGroupBox;
    Button3: TButton;
    LbVerlauf: TListBox;
    procedure BtCalcClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure LbVerlaufDblClick(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
    function Fakultaet(zahl: Integer):Extended;
    function IsInteger(Feld: TEdit):Boolean;
    function IsFloat(Feld: TEdit):Boolean;
    function CheckEingabe:Boolean;
    procedure showverlauf;
    procedure showhinweise;
    procedure addverlauf(vpz,vpn,vk,vn,vb: String);
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  pz: Extended;
  pn: Int64;
  k: Int64;
  n: Int64;
  matrix: Extended;
  matrixz: Extended;
  matrixn: Extended;

```

```

    faktor1: Extended;
    faktor2: Extended;
    exponent: Extended;
    b: Extended;
    ergebnis: Extended;
    overwrite: Boolean;

implementation

{$R *.dfm}

//Eigene Funktionen und Prozeduren
function TForm1.Fakultaet(zahl: Integer):Extended;
var i: Integer;
    fak: Extended;
begin
    fak:=1;
    for i:=1 to zahl do
        fak:=fak*i;
    result:=fak;
end;

function TForm1.IsInteger(Feld: TEdit):Boolean;
var i:Integer;
begin
    result:=true;
    for i:=1 to Length(Feld.Text) do
        case Feld.Text[i] of
            '0': result:=true;
            '1': result:=true;
            '2': result:=true;
            '3': result:=true;
            '4': result:=true;
            '5': result:=true;
            '6': result:=true;
            '7': result:=true;
            '8': result:=true;
            '9': result:=true;
        else
            begin
                result:=false;
                exit;
            end;
        end;
    end;
    if (Feld.Text = '') or (Length(Feld.Text) >9) then result:=false;
end;

function TForm1.IsFloat(Feld: TEdit):Boolean;
var i:Integer;
    komma: Integer;
    e: Integer;
    exponent: String;
    minus: Integer;
    pos: Integer;
begin
    result:=true;
    komma:=0;
    e:=0;
    minus:=0;
    for i:=1 to Length(Feld.Text) do
        case Feld.Text[i] of
            '0': result:=true;
            '1': result:=true;
            '2': result:=true;
            '3': result:=true;
            '4': result:=true;
            '5': result:=true;

```

```

        '6': result:=true;
        '7': result:=true;
        '8': result:=true;
        '9': result:=true;
        ',': begin result:=true; komma:=komma+1; end;
        'E': begin result:=true; e:=e+1; pos:=i+1 end;
        '-': begin result:=true; minus:=minus+1; pos:=i+1; end;
    else
    begin
        result:=false;
        exit;
    end;
end;

if (komma > 1) or (e > 1) or (minus > 1) then
begin
    result:=false;
    exit;
end;

if (e = 1) and (minus = 1) then
    if (Feld.Text[pos-2] <> 'e') and (Feld.Text[pos-2] <> 'E') then
    begin
        result:=false;
        exit;
    end;

if e = 1 then
begin
    exponent:=Copy(Feld.Text, pos, length(Feld.Text)-pos+1);
    if length(exponent) <= 4 then
        if (minus = 1) and (StrToInt(exponent) > 4950) then
        begin
            result:=false;
            exit;
        end
        else if (minus = 0) and (StrToInt(exponent) > 4931) then
        begin
            result:=false;
            exit;
        end;
    if length(exponent) > 4 then
    begin
        result:=false;
        exit;
    end;
end;

if (Feld.Text[1] = ',') or (Feld.Text[1] = 'E') or (Feld.Text[1] = '-')
then
begin
    result:=false;
    exit;
end;
if (Feld.Text = '') or (Length(Feld.Text) > 50) then result:=false;
end;

function TForm1.CheckEingabe:Boolean;
begin
    result:=true;

    if not IsFloat(Edpz) then
    begin
        showmessage('Fehler: Der Zähler von p ist keine gültige Zahl zwischen 1
und 500. ');
        result:=false;
    end;
end;

```

```

        edpz.SetFocus;
        exit;
    end
else if (IsFloat(Edpz)) then
    if StrToFloat(Edpz.Text) > 500 then
        begin
            showmessage('Fehler: Der Zähler von p ist keine gültige Zahl zwischen
1 und 500.');
```

1 und 500.');

```

            result:=false;
            edpz.SetFocus;
            exit;
        end;

    if not IsInteger(Edpn) then
        begin
            showmessage('Fehler: Der Nenner von p ist keine gültige natürliche Zahl
zwischen 1 und 500.');
```

zwischen 1 und 500.');

```

            result:=false;
            edpn.SetFocus;
            exit;
        end
    else if (IsInteger(Edpn)) then
        if StrToInt(Edpn.Text) > 500 then
            begin
                showmessage('Fehler: Der Nenner von p ist keine gültige natürliche
Zahl zwischen 1 und 500.');
```

Zahl zwischen 1 und 500.');

```

                result:=false;
                edpn.SetFocus;
                exit;
            end;

            if (IsInteger(Edpz)) and (IsInteger(Edpn)) then
                if (StrToFloat(Edpz.Text) / StrToFloat(Edpn.Text) < 0) or
(StrToFloat(Edpz.Text) / StrToFloat(Edpn.Text) > 1) then
                    begin
                        showmessage('Fehler: Die Wahrscheinlichkeit p liegt nicht zwischen 0
und 1.');
```

und 1.');

```

                        result:=false;
                        exit;
                    end;

            if not IsInteger(Edk) then
                begin
                    showmessage('Fehler: i ist keine gültige natürliche Zahl zwischen 1 und
1700.');
```

1700.');

```

                    result:=false;
                    edk.setfocus;
                    exit;
                end
            else if (IsInteger(Edk)) then
                if StrToInt(Edk.Text) > 1700 then
                    begin
                        showmessage('Fehler: i ist keine gültige natürliche Zahl zwischen 1
und 1700.');
```

und 1700.');

```

                        result:=false;
                        edk.setfocus;
                        exit;
                    end;

            if not IsInteger(Edn) then
                begin
                    showmessage('Fehler: n ist keine gültige natürliche Zahl zwischen 1 und
1700.');
```

1700.');

```

                    result:=false;
                    edn.setfocus;
                    exit;
                end
            end
        end
    end
end

```



```

    else if (IsInteger(Edn)) then
        if StrToInt(Edn.Text) > 1700 then
            begin
                showmessage('Fehler: n ist keine gültige natürliche Zahl zwischen 1
und 1700.');
```

result:=false;

edn.setfocus;

exit;

end;

```

    if (IsInteger(Edk)) and (IsInteger(Edn)) then
        if StrToInt(Edk.Text) > StrToInt(Edn.Text) then
            begin
                showmessage('Fehler: i > n');
```

result:=false;

edk.setfocus;

exit;

end;

end;

```

procedure TForm1.showverlauf;
begin
    Label4.Visible:=false;
    Label5.Visible:=false;
    Label6.Visible:=false;
    GroupBox3.Visible:=true;
end;

procedure TForm1.showhinweise;
begin
    Label4.Visible:=true;
    Label5.Visible:=true;
    Label6.Visible:=true;
    GroupBox3.Visible:=false;
end;

procedure TForm1.addverlauf(vpz,vpn,vk,vn,vb: String);
var zeile: String;
begin
    zeile:=('p=' + vpz + '/' + vpn + ';    i=' + vk + ';    n=' + vn + ';    B='
+ vb);
    LbVerlauf.Items.add(zeile);
end;

//Programm
procedure TForm1.BtCalcClick(Sender: TObject);
var i:Integer;
begin
    //B = matrix * p^k * (1-p)^(n-k)
    // p^k = faktor1, (1-p) = faktor2, (n-k) = exponent
    //Eingaben prüfen
    if not CheckEingabe then exit;
    //Datein einlesen
    pz:=StrToFloat(edpz.text);
    pn:=StrToInt(edpn.text);
    k:=StrToInt(edk.text);
    n:=StrToInt(edn.text);
    //Variable Ergebnis initialisieren
    ergebnis:=0;
    //Ergebnisse berechnen
    EdK6.Text:='';
    for i:=k to n do
        begin
            matrixz:=Fakultaet(n);
            EdK1.Text:='n! = ' + FloatToStr(Matrixz);
            EdK2.Text:='i! = ' + FloatToStr(Fakultaet(k)) + '; (n-i)! = ' +
FloatToStr(Fakultaet((n-k)));

```

```

matrixn:=Fakultaet(i)*Fakultaet((n-i));
EdK3.Text:='i! - (n-i)! = ' + FloatToStr(matrixn);
matrix:=matrixz / matrixn;
faktor1:=(Power(pz, i))/(Power(pn, i));
EdK4.Text:='p^i = ' + FloatToStr((Power(pz, k))/(Power(pn, k)));
faktor2:=((1 * pn - pz) / pn);
EdK5.Text:='1-p = ' + FloatToStr(faktor2);
Exponent:=n - i;
B:=matrix * faktor1 * (Power(faktor2, Exponent));
EdK6.Text:= EdK6.Text + 'P(' + IntToStr(i) + ') = ' + FloatToStrF(B,
ffixed, 10,5) + '; ';
ergebnis:=ergebnis+b;
end;
showmessage(FloatToStr(Ergebnis));
EdErgebnis.Text:='B=' + FloatToStr(Ergebnis);
showverlauf;
addverlauf(FloatToStr(pz), IntToStr(pn), IntToStr(k), IntToStr(n),
FloatToStrF(ergebnis, ffixed, 1000,5));
edpz.Text:=FloatToStr(ergebnis);
edpn.Text:='1';
edk.Text:='';
edk.SetFocus;
edn.Text:='';
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
    showhinweise;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
    showverlauf;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
    showverlauf;
    overwrite:=false;
end;

procedure TForm1.LbVerlaufDb1Click(Sender: TObject);
begin
    Showmessage(LbVerlauf.Items[LbVerlauf.Itemindex])
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
    If (FileExists('boegen-verlauf.txt')) and (overwrite = false) then
    begin
        showmessage('Warnung: Eine Datei "boegen-verlauf.txt" existiert
bereits. Diese wird überschrieben, wenn nochmal auf "Verlauf speichern"
geklickt wird!');
        overwrite:=true;
        exit;
    end
    else
    begin
        LbVerlauf.Items.SaveToFile('boegen-verlauf.txt');
        showmessage('Der Verlauf wurde in der Datei "boegen-verlauf.txt"
gespeichert. ');
        overwrite:=false;
    end;
end;

end.

```

Optimaler_ankreuzbogen.exe:

The screenshot shows the 'Der optimale Ankreuzbogen' window with the 'Idealen Test entwerfen' tab selected. The window has a title bar with standard Windows controls. Below the title bar, there are two radio buttons: 'Idealen Test entwerfen' (selected) and 'Fertigen Test überprüfen'. Below these, there are two more radio buttons: '... einer richtigen Antwort' and '... mehreren richtigen Antworten' (selected). The main area is titled 'Einen idealen Test entwerfen mit...'. It contains several input fields and buttons. 'Richtige Antworten je Frage:' has a dropdown menu set to '[mehrere]' and an 'Erstellen!' button. 'Zeitaufwand Fragen zu Antworten:' has two input fields, '1,5' and '0,6', with 'zu' between them, and a 'Hinweise zeigen' button. 'Bearbeitungszeit (in Minuten):' has an input field with '20' and two checkboxes: 'Debug-Modus' and 'Gegenvorschlag'. Below these are three rows of text: 'Anzahl der Fragen (x): 6 ...', 'Wahrscheinlichkeit (p): 0,00000 ...', and 'Anzahl der Antwortmöglichkeiten (z): 3 ...'. At the bottom, there is a status bar that says 'Debug-Modus deaktiviert'.

Der optimale Ankreuzbogen

☒ Idealen Test entwerfen ☐ Fertigen Test überprüfen

☐ ... einer richtigen Antwort ☒ ... mehreren richtigen Antworten

Einen idealen Test entwerfen mit...

Richtige Antworten je Frage: [mehrere] **Erstellen!**

Zeitaufwand Fragen zu Antworten: **Hinweise zeigen**

1,5 zu 0,6 ☐ Debug-Modus

Bearbeitungszeit (in Minuten): 20 ☐ Gegenvorschlag

Anzahl der Fragen (x): 6 ...

Wahrscheinlichkeit (p): 0,00000 ...

Anzahl der Antwortmöglichkeiten (z): 3 ...

Debug-Modus deaktiviert

Screenshot: Erstellung eines optimalen Tests

The screenshot shows the 'Der optimale Ankreuzbogen' window with the 'Fertigen Test überprüfen' tab selected. The window has a title bar with standard Windows controls. Below the title bar, there are two radio buttons: 'Idealen Test entwerfen' and 'Fertigen Test überprüfen' (selected). Below these, there are two more radio buttons: '... einer richtigen Antwort' and '... mehreren richtigen Antworten' (selected). The main area is titled 'Einen fertigen Test überprüfen'. It contains several input fields and buttons. 'Richtige Antworten je Frage:' has a dropdown menu set to '[mehrere]'. 'Anzahl der Fragen:' has an input field with '10' and an 'Überprüfen!' button. 'Anzahl der Antwortmöglichkeiten:' has an input field with '3' and a checkbox for 'Debug-Modus'. Below these are two rows of text: 'Wahrscheinlichkeit (p): 0,00000' and 'Dieser Test ist: sehr gut'. At the bottom, there is a status bar that says 'Debug-Modus deaktiviert'.

Der optimale Ankreuzbogen

☐ Idealen Test entwerfen ☒ Fertigen Test überprüfen

☐ ... einer richtigen Antwort ☒ ... mehreren richtigen Antworten

Einen fertigen Test überprüfen

Richtige Antworten je Frage: [mehrere]

Anzahl der Fragen: 10 **Überprüfen!**

Anzahl der Antwortmöglichkeiten: 3 ☐ Debug-Modus

Wahrscheinlichkeit (p): 0,00000

Dieser Test ist: sehr gut

Debug-Modus deaktiviert

Screenshot: Überprüfung eines fertigen Tests

```

unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms,
  Dialogs, StdCtrls, Math, ExtCtrls;

type
  TForm1 = class(TForm)
    GroupBox1: TGroupBox;
    Label1: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Lx: TLabel;
    Lp: TLabel;
    Lz: TLabel;
    Edb: TEdit;
    Edt: TEdit;
    BtCalc: TButton;
    CheckBox1: TCheckBox;
    Eda: TEdit;
    EdDebug: TEdit;
    GroupBox2: TGroupBox;
    Label2: TLabel;
    Label7: TLabel;
    Edx: TEdit;
    Label8: TLabel;
    Edz: TEdit;
    Label10: TLabel;
    Lp2: TLabel;
    BtCheck: TButton;
    Label12: TLabel;
    Lq: TLabel;
    CheckBox2: TCheckBox;
    CheckBox3: TCheckBox;
    GroupBox3: TGroupBox;
    Rb1: TRadioButton;
    Rb2: TRadioButton;
    GroupBox4: TGroupBox;
    Rb1AW: TRadioButton;
    RbMAw: TRadioButton;
    Lxg: TLabel;
    Lpg: TLabel;
    Lzg: TLabel;
    Label9: TLabel;
    LAw: TLabel;
    Button1: TButton;
    GroupBox5: TGroupBox;
    MHinweise: TMemo;
    Button2: TButton;
    Label11: TLabel;
    LAw2: TLabel;
    GroupBox6: TGroupBox;
    Label13: TLabel;
    EdpFrage: TEdit;
    Label15: TLabel;
    BtpFrage: TButton;
    Label16: TLabel;
    Label17: TLabel;
    Label18: TLabel;
    Label19: TLabel;
  end;

```

```

Shape1: TShape;
Label14: TLabel;
Label20: TLabel;
Label21: TLabel;
Label22: TLabel;
Label23: TLabel;
Label24: TLabel;
Edn1: TEdit;
Edn2: TEdit;
Edn3: TEdit;
Edn4: TEdit;
Edn5: TEdit;
Edn6: TEdit;
Lrel1: TLabel;
Lrel2: TLabel;
Lrel3: TLabel;
Lrel4: TLabel;
Lrel5: TLabel;
Lrel6: TLabel;
Lpn1: TLabel;
Lpn2: TLabel;
Lpn4: TLabel;
Lpn3: TLabel;
Lpn6: TLabel;
Lpn5: TLabel;
Timer1: TTimer;
LpFrageErgebnis: TLabel;
Label25: TLabel;
Edaktiv: TEdit;
BtHinweise: TButton;
GroupBox7: TGroupBox;
MHinweise2: TMemo;
BtCloseHinweise: TButton;
procedure BtCalcClick(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure CheckBox1Click(Sender: TObject);
procedure Rb1Click(Sender: TObject);
procedure Rb2Click(Sender: TObject);
procedure BtCheckClick(Sender: TObject);
procedure EdaClick(Sender: TObject);
procedure EdbClick(Sender: TObject);
procedure EdtClick(Sender: TObject);
procedure CheckBox2Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure EdpFrageChange(Sender: TObject);
procedure BtpFrageClick(Sender: TObject);
procedure Timer1Timer(Sender: TObject);
procedure EdaChange(Sender: TObject);
procedure Rb1AWClick(Sender: TObject);
procedure RbMAWClick(Sender: TObject);
procedure EdxChange(Sender: TObject);
procedure BtCloseHinweiseClick(Sender: TObject);
procedure BtHinweiseClick(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
function abl(x: Extended):Extended;
function abl2(x:Extended):Extended;
function fkt(x,anzz: Extended):Extended;
function z(x:Extended):Extended;
function newton(x:Extended):Extended;
function anzfrage:Extended;
function IsInteger(Feld:TEdit):Boolean;
function IsFloat(Feld: TEdit):Boolean;
function CheckEingabe:Boolean;

```

```

    procedure calc(opt:Integer);
    procedure NoTest;
    function CheckResult(x,anzz:Integer; wahrsch: Extended):Boolean;
    procedure EnableBenotung;
    procedure DisableBenotung;
    function bernoulli(wahrscheinlichkeit: Extended; von,
bis:Integer):Extended;
    function fakultaet(zahl:Integer):Extended;
    end;

var
    Form1: TForm1;
    a:Extended;
    b:Extended;
    t:Integer;
    x2:Integer;
    z2:Integer;
    maxdef:Extended;
    debug:boolean;
    status:boolean;
    original:boolean;
    gegenvorschlag:boolean;

implementation

{$R *.dfm}

//Eigene Prozeduren

//Rb1Aw: Eine Antwort je Frage ist richtig
//RbMAw: Mehrere Antworten je Frage sind richtig
//Rb1: Einen Test entwerfen
//Rb2: Einen Test überprüfen

function TForm1.abl(x:Extended):Extended;
var term1, arg:Extended;
begin
    term1:=t/(t-a*x);
    arg:=t/(b*x) - a/b;
    result:=term1 - LN(arg);
end;

function TForm1.abl2(x:Extended):Extended;
var zaehler:Extended;
    nenner:Extended;
begin
    zaehler:=t*t;
    nenner:=x*Power((a*x-t), 2);
    result:=zaehler/nenner;
end;

function TForm1.fkt(x,anzz: Extended):Extended;
var exponent:Extended;
    bruch:Extended;
begin
    if Rb1Aw.Checked = true then
    begin
        //eine richtige Antwort:  $p(x)=1/z^x$ 
        if anzz > 0 then result:=1/Power(anzz, x)
        else
            begin
                result:=1;
                status:=false;
            end;
    end;
end;

if RbMAw.Checked = true then

```

```

begin
  //mehrere richtige Antworten:  $p(x)=1/2^{(xz)}$ 
  result:=1/Power(2, anzz*x);
end;
if debug then EdDebug.Text:=EdDebug.Text + 'p='+FloatToStr(result)+'; ';
end;

function TForm1.z(x:Extended):Extended;
begin
  if x > 0 then result:=(t-a*x)/(b*x)
  else
    begin
      status:=false;
      result:=1;
      exit;
    end;
  if debug then EdDebug.Text:=EdDebug.Text + 'z=' + FloatToStr(result) + ';
';
end;

function TForm1.newton(x:Extended):Extended;
var i:Integer;
begin
  i:=0;
  repeat
    x:=x-(abl(x)/abl2(x));
    if x >= maxdef then x:=maxdef;
    if debug then Form1.Caption:=FloatToStr(abs(abl(x))) + ' --- ' +
IntToStr(i) + ' --- ' + FloatToStr(x);
    i:=i+1;
  until (abs(abl(x)) < (1/1000000)) or (i = 1000);
  result:=x;
  if debug then EdDebug.Text:=EdDebug.Text + 'x=' + FloatToStr(result)+ ';
';
end;

function TForm1.anzfrage:Extended;
begin
  result:=t/(2*a);
  if debug then EdDebug.Text:=EdDebug.Text + 'x=' + FloatToStr(result)+ ';
';
end;

function TForm1.IsInteger(Feld: TEdit):Boolean;
var i:Integer;
begin
  result:=true;
  for i:=1 to Length(Feld.Text) do
    case Feld.Text[i] of
      '0': result:=true;
      '1': result:=true;
      '2': result:=true;
      '3': result:=true;
      '4': result:=true;
      '5': result:=true;
      '6': result:=true;
      '7': result:=true;
      '8': result:=true;
      '9': result:=true;
    else
      begin
        result:=false;
        exit;
      end;
    end;
  end;
  if (Feld.Text = '') or (Length(Feld.Text) >9) then result:=false;
end;

```

```

function TForm1.IsFloat(Feld: TEdit):Boolean;
var i:Integer;
    komma: Integer;
    e: Integer;
    exponent: String;
    minus: Integer;
    pos: Integer;
begin
    result:=true;
    komma:=0;
    e:=0;
    minus:=0;
    pos:=0;
    for i:=1 to Length(Feld.Text) do
        case Feld.Text[i] of
            '0': result:=true;
            '1': result:=true;
            '2': result:=true;
            '3': result:=true;
            '4': result:=true;
            '5': result:=true;
            '6': result:=true;
            '7': result:=true;
            '8': result:=true;
            '9': result:=true;
            ',': begin result:=true; komma:=komma+1; pos:=i; end;
            else
                begin
                    result:=false;
                    exit;
                end;
            end;
        end;

    if (komma > 1) or (e > 1) or (minus > 1) then
    begin
        result:=false;
        exit;
    end;

    if Feld.Text[1] = ',' then
    begin
        result:=false;
        exit;
    end;
    if (Feld.Text = '') or (pos > 360) then result:=false;
end;

function TForm1.CheckEingabe:Boolean;
begin
    result:=true;
    if Rb1.Checked = true then
    begin

        if not IsFloat(Eda) then
        begin
            showmessage('Fehler: Der Parameter a ist keine gültige reelle Zahl');
            result:=false;
            exit;
        end;

        if not (IsFloat(Edb)) then
        begin
            showmessage('Fehler: Der Parameter b ist keine gültige reelle Zahl');
            result:=false;
        end;
    end;
end;

```



```

        exit;
    end;

    if not IsInteger(Edt) then
    begin
        showmessage('Fehler: Der Wert von t ist keine gültige natürliche
Zahl');
        result:=false;
        exit;
    end;

    //Allgemein
    if (IsFloat(Eda)) and (IsFloat(Edb)) and (IsInteger(Edt)) then
        if (StrToFloat(Eda.Text) <=0.5) or (StrToFloat(Edb.Text) <= 0.5) {or
(StrToFloat(Eda.Text) > (StrToInt(Edt.Text)/10)) or (StrToFloat(Edb.Text) >
(StrToInt(Edt.Text)/10))} or (StrToFloat(Eda.Text) > StrToInt(Edt.Text)/5)
or (StrToFloat(Edb.Text) > StrToInt(Edt.Text)/5) then
        begin
            showmessage('Fehler: Die Parameter a und b müssen zwischen 0,5 und
t/5 liegen!');
            result:=false;
            exit;
        end
        else if (StrToFloat(Eda.Text) >= StrToInt(Edt.Text)) then
        begin
            showmessage('Fehler: Der Parameter t muss größer als der Parameter
a sein!');
            result:=false;
            exit;
        end;

        if (IsInteger(Edt)) then
            if (StrToInt(Edt.Text) > 360) or (StrToInt(Edt.Text) <= 5) then
            begin
                showmessage('Fehler: Der Parameter t muss größer als 5 darf nicht
größer als 360 sein!');
                result:=false;
                exit;
            end;

            //Eine richtige Antwort
            if (Rb1Aw.Checked = true) and (IsFloat(Eda)) and (IsInteger(Edt)) and
(IsFloat(Edb)) then
                if StrToInt(Edt.Text) > StrToFloat(Eda.Text) then
                begin
                    if LN((StrToInt(Edt.Text) -
StrToFloat(Eda.Text))/StrToFloat(Edb.Text)) <=
StrToInt(Edt.Text)/(StrToInt(Edt.Text) - StrToFloat(Eda.Text)) then
                        begin
                            showmessage('Fehler: LN-Fehler! a sollte sehr klein zu t sein,
und b sollte ebenfalls klein gewählt werden.');
```

```

begin
    showmessage('Fehler: Der Wert von z ist keine gültige natürliche
Zahl');
    result:=false;
    exit;
end;

end;
end;

procedure TForm1.calc(opt:Integer);
var gx:Integer;
    gp:Extended;
    ok:Boolean;
begin
    //Optionen:
    //0: Normale Berechnung
    //1: Gegenvorschlag machen
    //2: Nur Gegenvorschlag machen

    //eine richtige Antwort

    ok:=true;

    if Rb1Aw.Checked = true then
    begin
        if (opt = 0) or (opt = 1) then
        begin
            Lx.Caption:=FloatToStr(Int(newton(1)));
            Lz.Caption:=FloatToStr(round(z(Int(newton(1)))));
            Lp.Caption:=FloatToStrF(fkt(Int(newton(1)),
round(z(Int(newton(1))))), ffexponent, 3,0);
        end;
        //Gegenvorschlag
        if (opt = 1) or (opt = 2) then
        begin
            if debug then EdDebug.Text:=EdDebug.Text + '- - - Gegenvorschlag - -
-; ';
            Lzg.Caption:=IntToStr(5);
            gx:=round(t/(a+b*5));
            if gx > maxdef then ok:=false;
            if ok = true then gp:=fkt(gx, 5);
            if (CheckResult(gx, 5, gp) = false) or (gp > 1/4) or (ok = false)
then
            begin
                Lxg.Caption:='- - -';
                Lpg.Caption:='- - -';
                Lzg.Caption:='- - -';
                showmessage('Es ist leider kein realistischer Gegenvorschlag
möglich. Bitte die Parameter a, b und t sinnvoll wählen.');
```

möglich. Bitte die Parameter a, b und t sinnvoll wählen.');

```

                gegenvorschlag:=false;
                DisableBenotung;
                exit;
            end;
            Lxg.Caption:=IntToStr(gx);
            Lpg.Caption:=FloatToStrF(gp, ffexponent, 3,0);
            EnableBenotung;
            gegenvorschlag:=true;
        end;
    end;

    //mehrere richtige Antworten
    if RbMAw.Checked = true then
    begin
        if (opt = 0) or (opt = 1) then
        begin
            Lx.Caption:=FloatToStr(Int(anzfrage));

```

```

        Lz.Caption:=FloatToStr(round(z(Int(anzfrage))));
        Lp.Caption:=FloatToStrF(fkt(Int(anzfrage), round(z(Int(anzfrage)))),
ffexponent, 3,0);
    end;
    //Gegenvorschlag
    if (opt = 1) or (opt = 2) then
    begin
        if debug then EdDebug.Text:=EdDebug.Text + '- - - Gegenvorschlag - -
-; ';
        Lzg.Caption:=IntToStr(5);
        gx:=round(t/(a+b*5));
        if (gx > maxdef) then ok:=false;
        if ok = true then gp:=fkt(gx, 5);
        if (CheckResult(gx, 5, gp) = false) or (gp > 1/4) or (ok = false)
then
            begin
                Lxg.Caption:='- - -';
                Lpg.Caption:='- - -';
                Lzg.Caption:='- - -';
                showmessage('Es ist leider kein realistischer Gegenvorschlag
möglich. Bitte die Parameter a, b und t sinnvoll wählen.');

```

```

BtpFrage.Enabled:=true;
Edn1.Text:='';
Edn2.Text:='';
Edn3.Text:='';
Edn4.Text:='';
Edn5.Text:='';
Edn6.Text:='';
Edn1.Enabled:=true;
Edn2.Enabled:=true;
Edn3.Enabled:=true;
Edn4.Enabled:=true;
Edn5.Enabled:=true;
Edn6.Enabled:=true;
Lrel1.Caption:='...%';
Lrel2.Caption:='...%';
Lrel3.Caption:='...%';
Lrel4.Caption:='...%';
Lrel5.Caption:='...%';
Lrel6.Caption:='...%';
Lpn1.Caption:='...';
Lpn2.Caption:='...';
Lpn3.Caption:='...';
Lpn4.Caption:='...';
Lpn5.Caption:='...';
Lpn6.Caption:='...';
Edaktiv.Text:='Ja';
Edaktiv.Color:=clgreen;
Timer1.Enabled:=true;
end;

```

```

procedure TForm1.DisableBenotung;
begin
  EdpFrage.Text:='';
  LpFrageErgebnis.Caption:='';
  EdpFrage.Enabled:=false;
  BtpFrage.Enabled:=false;
  Edn1.Text:='';
  Edn2.Text:='';
  Edn3.Text:='';
  Edn4.Text:='';
  Edn5.Text:='';
  Edn6.Text:='';
  Edn1.Enabled:=false;
  Edn2.Enabled:=false;
  Edn3.Enabled:=false;
  Edn4.Enabled:=false;
  Edn5.Enabled:=false;
  Edn6.Enabled:=false;
  Lrel1.Caption:='...%';
  Lrel2.Caption:='...%';
  Lrel3.Caption:='...%';
  Lrel4.Caption:='...%';
  Lrel5.Caption:='...%';
  Lrel6.Caption:='...%';
  Lpn1.Caption:='...';
  Lpn2.Caption:='...';
  Lpn3.Caption:='...';
  Lpn4.Caption:='...';
  Lpn5.Caption:='...';
  Lpn6.Caption:='...';
  Edaktiv.Text:='Nein';
  Edaktiv.Color:=clred;
  Timer1.Enabled:=false;
end;

```

```

function TForm1.bernoulli(wahrscheinlichkeit:Extended; von,
bis:Integer):Extended;

```

```

var i:Integer;
    B:Extended;
    ergebnis:Extended;
    matrixz:Extended;
    matrixn:Extended;
    matrix:Extended;
    faktor1:Extended;
    faktor2:Extended;
    exponent:Extended;
begin
//W = matrix * p^k * (1-p)^(n-k)
    // p^k = faktor1, (1-p) = faktor2, (n-k) = exponent

    //Variable Ergebnis initialisieren
    ergebnis:=0;
//Ergebnisse berechnen
    for i:= von to bis do
        begin
            matrixz:=Fakultaet(bis);
            matrixn:=Fakultaet(i)*Fakultaet((bis-i));
            matrix:=matrixz / matrixn;
            faktor1:=(Power(wahrscheinlichkeit, i));
            faktor2:=(1 - wahrscheinlichkeit);
            Exponent:=bis - i;
            B:=matrix * faktor1 * (Power(faktor2, Exponent));
            ergebnis:=ergebnis+B;
        end;
        result:=ergebnis;
    end;

function TForm1.Fakultaet(zahl: Integer):Extended;
var i: Integer;
    fak: Extended;
begin
    fak:=1;
    for i:=1 to zahl do
        fak:=fak*i;
    result:=fak;
end;

//Programm Prozeduren
//#####
procedure TForm1.BtCalcClick(Sender: TObject);
var i: Extended;
    tmp: Extended;
begin
    DisableBenotung;
    original:=false;
    gegenvorschlag:=false;
    status:=true;
    Lx.Caption:='...';
    Lp.Caption:='...';
    Lz.Caption:='...';
    Lxg.Caption:='...';
    Lpg.Caption:='...';
    Lzg.Caption:='...';
    If not CheckEingabe then exit;
    a:=StrToFloat(Eda.Text);
    b:=StrToFloat(Edb.Text);
    t:=StrToInt(Edt.Text);

    maxdef:=(t/a)-0.1;
    if debug then EdDebug.Text:='';
    Lx.Caption:='';
    Lp.Caption:='';
    Lz.Caption:='';

```

```

    If Rb1Aw.Checked = true then LAw.Caption:='[ eine ]'
    else LAw.Caption:='[ mehrere ]';

    calc(0);
    if (status = false) or (CheckResult(StrToInt(Lx.Caption),
StrToInt(Lz.Caption), StrToFloat(Lp.Caption)) = false) then
        NoTest
    else
        begin
            EnableBenotung;
            original:=true;
        end;

    if Lx.Caption <> '- - -' then
    begin
        if StrToInt(Lx.Caption) < 3 then showmessage(Lx.Caption + ' Fragen sind
unrealistisch! Bitte die Parameter a und b sinnvoll wählen!');
        if (StrToInt(Lz.Caption) > 5) or (StrToInt(Lz.Caption) < 2) then
showmessage(Lz.Caption + ' Antwortmöglichkeiten sind unrealistisch! Bitte
die Paramter a und b sinnvoll wählen!');
        if ((StrToInt(Lz.Caption) > 5) or (StrToInt(Lz.Caption) < 2) or
(StrToInt(Lx.Caption) < 3)) and (CheckBox3.Checked = true) then calc(1);
        end;
    end;

procedure TForm1.FormCreate(Sender: TObject);
begin
    debug:=false;
    DisableBenotung;
    original:=false;
    gegenvorschlag:=false;
end;

procedure TForm1.CheckBox1Click(Sender: TObject);
begin
    if CheckBox1.Checked = true then
    begin
        debug:=true;
        EdDebug.Text:=''
    end
    else
    begin
        debug:=false;
        EdDebug.Text:='Debug-Modus deaktiviert';
        Form1.Caption:='Der optimale Ankreuzbogen';
    end;
end;

procedure TForm1.Rb1Click(Sender: TObject);
begin
    GroupBox1.Visible:=true;
    GroupBox2.Visible:=false;
    GroupBox5.Visible:=false;
    DisableBenotung;

    if CheckBox1.Checked = true then
    begin
        debug:=true;
        EdDebug.Text:=''
    end
    else
    begin
        debug:=false;
        EdDebug.Text:='Debug-Modus deaktiviert';
        Form1.Caption:='Der optimale Ankreuzbogen';
    end;
end;

```

```

    end;
end;

procedure TForm1.Rb2Click(Sender: TObject);
begin
    GroupBox1.Visible:=false;
    GroupBox2.Visible:=true;
    GroupBox5.Visible:=false;
    DisableBenotung;

    if CheckBox2.Checked = true then
    begin
        debug:=true;
        EdDebug.Text:='';
    end
    else
    begin
        debug:=false;
        EdDebug.Text:='Debug-Modus deaktiviert';
        Form1.Caption:='Der optimale Ankreuzbogen';
    end;

end;

procedure TForm1.BtCheckClick(Sender: TObject);
var exponent:Extended;
    erg:Extended;
begin
    DisableBenotung;
    If not CheckEingabe then exit;
    x2:=StrToInt(Edx.Text);
    z2:=StrToInt(Edz.Text);
    if Rb1Aw.Checked = true then erg:=1/Power(z2, x2)
    else if RbMAw.Checked = true then erg:=1/Power(2,x2*z2);
    if debug then EdDebug.Text:=FloatToStr(erg);
    Lp2.Caption:=FloatToStrF(erg, ffexponent, 3,0);
    If Rb1Aw.Checked = true then LAw2.Caption:='[ eine ]'
    else if RbMAw.Checked = true then LAw2.Caption:='[ mehrere ]';

    //Einschätzung abgeben:
    if abs(erg) <= 1/100000 then Lq.Caption:='sehr gut'
    else if (abs(erg) <= 1/10000) and (abs(erg) > 1/100000) then
Lq.Caption:='gut'
    else if (abs(erg) <= 1/1000) and (abs(erg) > 1/10000) then
Lq.Caption:='akzeptabel'
    else if (abs(erg) <= 1/100) and (abs(erg) > 1/1000) then
Lq.Caption:='noch akzeptabel'
    else if (abs(erg) <= 1/10) and (abs(erg) > 1/100) then
Lq.Caption:='schlecht'
    else if (abs(erg) <= 1/4) and (abs(erg) > 1/10) then Lq.Caption:='sehr
schlecht'
    else if (abs(erg) > 1/4) then Lq.Caption:='unbrauchbar';
    EnableBenotung;
end;

procedure TForm1.EdaClick(Sender: TObject);
begin
    if Eda.Text = 'a' then Eda.Text:=''
end;

procedure TForm1.EdbClick(Sender: TObject);
begin
    if Edb.text = 'b' then Edb.Text:=''
end;

procedure TForm1.EdtClick(Sender: TObject);
begin

```

```

    if Edt.Text = 't' then Edt.Text:=''
end;

procedure TForm1.CheckBox2Click(Sender: TObject);
begin
    if CheckBox2.Checked = true then
    begin
        debug:=true;
        EdDebug.Text:=''
    end
    else
    begin
        debug:=false;
        EdDebug.Text:='Debug-Modus deaktiviert';
        Form1.Caption:='Der optimale Ankreuzbogen';
    end;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
    GroupBox1.Visible:=true;
    GroupBox5.Visible:=false;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
    GroupBox5.Visible:=true;
    GroupBox1.Visible:=false;
end;

procedure TForm1.EdpFrageChange(Sender: TObject);
begin
    BtpFrage.Default:=true;
    BtCalc.Default:=false;
    BtCheck.Default:=false;
end;

procedure TForm1.BtpFrageClick(Sender: TObject);
{Diese Funktion zu dieser Prozedur
befindet sich noch in der Testphase und kann daher
instabil sein!
}
var pFrage:Extended;
    tmpergebnis:Extended;
    gesFragen:Integer;
    anzantworten:Integer;
begin
    //Entwurfener Test
    if Rb1.Checked = true then
    begin
        //Nur Originalvorschlag vorhanden
        if (original = true) and (gegenvorschlag = false) then
        begin
            gesFragen:=StrToInt(Lx.Caption);
            anzantworten:=StrToInt(Lz.Caption);
        end
        //Gegenvorschlag vorhanden
        else if gegenvorschlag = true then
        begin
            gesFragen:=StrToInt(Lxg.Caption);
            anzantworten:=StrToInt(Lzg.Caption);
        end;
        //Eingabe prüfen
        if IsInteger(EdpFrage) then
            if StrToInt(EdpFrage.Text) > gesFragen then
                begin

```



```

        showmessage('Es können nicht mehr Fragen als vorhanden beantwortet
werden!');
        exit;
    end;
end
//Fertiger Test
else if Rb2.Checked = true then
begin
    if (IsInteger(Edx)) and (IsInteger(Edz)) then
    begin
        gesFragen:=StrToInt(Edx.Text);
        anzantworten:=StrToInt(Edz.Text);
    end;
    //Eingaben prüfen
    if IsInteger(EdpFrage) then
        if StrToInt(EdpFrage.Text) > gesFragen then
        begin
            showmessage('Es können nicht mehr Fragen als vorhanden beantwortet
werden!');
            exit;
        end;
    end;

    //Eine richtige Antwort
    if Rb1Aw.Checked = true then
    begin
        pFrage:=1/anzantworten;
        //Bernoulli-Kette anwenden, ergebnis anzeigen
        LpFrageErgebnis.Caption:=FloatToStrF(bernoulli(pFrage,
StrToInt(EdpFrage.Text), gesFragen), ffexponent, 10,0);
    end;

    //Mehrere richtigen Antworten
    if RbMAw.Checked = true then
    begin
        pFrage:=1/Power(2,anzantworten);
        //Bernoulli-Kette anwenden
        LpFrageErgebnis.Caption:=FloatToStrF(bernoulli(pFrage,
StrToInt(EdpFrage.Text), gesFragen), ffexponent, 10,0);
    end;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
{Diese Funktion zu dieser Prozedur
befindet sich noch in der Testphase und kann daher
instabil sein!
}
var pFrage:Extended;
    tmpergebnis:Extended;
    gesFragen:Integer;
    anzantworten:Integer;
begin
    if Rb1.Checked = true then
    begin
        if (original = true) and (gegenvorschlag = false) then
        begin
            gesFragen:=StrToInt(Lx.Caption);
            anzantworten:=StrToInt(Lz.Caption);
        end
        else if (gegenvorschlag = true) then
        begin
            gesFragen:=StrToInt(Lxg.Caption);
            anzantworten:=StrToInt(Lzg.Caption);
        end;
    end
    else if Rb2.Checked = true then

```

```

    if (IsInteger(Edx)) and (IsInteger(Edz)) then
    begin
        gesFragen:=StrToInt(Edx.Text);
        anzahlworten:=StrToInt(Edz.Text);
    end;

    //Eine richtige Antwort
    if RblAw.Checked = true then
    begin
        pFrage:=1/anzahlworten;
    end;

    //Mehrere richtigen Antworten
    if RbMAw.Checked = true then
    begin
        pFrage:=1/Power(2,anzahlworten);
    end;

    //Prozentzahlen und Wahrscheinlichkeiten berechnen

    if IsInteger(Edn1) then
    begin
        if StrToInt(Edn1.Text) <= gesFragen then
        begin
            Lrel1.Caption:=IntToStr(round(100*StrToInt(Edn1.Text)/gesFragen)) +
'%';
            //Bernoulli anwenden
            Lpn1.Caption:=FloatToStrF(bernoulli(pFrage, StrToInt(Edn1.Text),
StrToInt(Edn1.Text)), ffexponent, 5,0);
        end
        else
        begin
            Lrel1.Caption:='...%';
            Lpn1.Caption:='...';
        end;
    end
    else
    begin
        Lrel1.Caption:='...%';
        Lpn1.Caption:='...';
    end;

    if IsInteger(Edn2) then
    begin
        if StrToInt(Edn2.Text) <= gesFragen then
        begin
            Lrel2.Caption:=IntToStr(round(100*StrToInt(Edn2.Text)/gesFragen)) +
'%';
            //Bernoulli anwenden
            if IsInteger(Edn1) then Lpn2.Caption:=FloatToStrF(bernoulli(pFrage,
StrToInt(Edn2.Text), StrToInt(Edn1.Text) - 1), ffexponent, 5,0);
        end
        else
        begin
            Lrel2.Caption:='...%';
            Lpn2.Caption:='...';
        end;
    end
    else
    begin
        Lrel2.Caption:='...%';
        Lpn2.Caption:='...';
    end;

    if IsInteger(Edn3) then
    begin
        if StrToInt(Edn3.Text) <= gesFragen then

```

```

begin
  Lrel3.Caption:=IntToStr(round(100*StrToInt(Edn3.Text)/gesFragen)) +
'%';
  //Bernoulli anwenden;
  if IsInteger(Edn2) then Lpn3.Caption:=FloatToStrF(bernoulli(pFrage,
StrToInt(Edn3.Text), StrToInt(Edn2.Text) - 1), ffexponent, 5,0);
end
  else
    begin
      Lrel3.Caption:='...%';
      Lpn3.Caption:='...';
    end;
end
  else
    begin
      Lrel3.Caption:='...%';
      Lpn3.Caption:='...';
    end;

if IsInteger(Edn4) then
begin
  if StrToInt(Edn4.Text) <= gesFragen then
begin
  Lrel4.Caption:=IntToStr(round(100*StrToInt(Edn4.Text)/gesFragen)) +
'%';
  //Bernoulli anwenden;
  if IsInteger(Edn3) then Lpn4.Caption:=FloatToStrF(bernoulli(pFrage,
StrToInt(Edn4.Text), StrToInt(Edn3.Text) - 1), ffexponent, 5,0);
end
  else
    begin
      Lrel4.Caption:='...%';
      Lpn4.Caption:='...';
    end;
end
  else
    begin
      Lrel4.Caption:='...%';
      Lpn4.Caption:='...';
    end;

if IsInteger(Edn5) then
begin
  if StrToInt(Edn5.Text) <= gesFragen then
begin
  Lrel5.Caption:=IntToStr(round(100*StrToInt(Edn5.Text)/gesFragen)) +
'%';
  //Bernoulli anwenden;
  if IsInteger(Edn4) then Lpn5.Caption:=FloatToStrF(bernoulli(pFrage,
StrToInt(Edn5.Text), StrToInt(Edn4.Text) - 1), ffexponent, 5,0);
end
  else
    begin
      Lrel5.Caption:='...%';
      Lpn5.Caption:='...';
    end;
end
  else
    begin
      Lrel5.Caption:='...%';
      Lpn5.Caption:='...';
    end;

if IsInteger(Edn6) then
begin
  if StrToInt(Edn6.Text) <= gesFragen then
begin

```

```

        Lrel6.Caption:=IntToStr(round(100*StrToInt(Edn6.Text)/gesFragen)) +
'%';
        //Bernoulli anwenden;
        if IsInteger(Edn5) then Lpn6.Caption:=FloatToStrF(bernoulli(pFrage,
StrToInt(Edn6.Text), StrToInt(Edn5.Text) - 1), ffexponent, 5,0);
        end
        else
        begin
            Lrel6.Caption:='...%';
            Lpn6.Caption:='...';
        end;
    end
    else
    begin
        Lrel6.Caption:='...%';
        Lpn6.Caption:='...';
    end;
end;

end;

procedure TForm1.EdaChange(Sender: TObject);
begin
    BtCalc.Default:=true;
    BtCheck.Default:=false;
    BtpFrage.Default:=false;
end;

procedure TForm1.Rb1AWClick(Sender: TObject);
begin
    DisableBenotung;
end;

procedure TForm1.RbMAWClick(Sender: TObject);
begin
    DisableBenotung;
end;

procedure TForm1.EdxChange(Sender: TObject);
begin
    BtpFrage.Default:=false;
    BtCalc.Default:=false;
    BtCheck.Default:=true;
end;

procedure TForm1.BtCloseHinweiseClick(Sender: TObject);
begin
    GroupBox6.Visible:=true;
    GroupBox7.Visible:=false;
end;

procedure TForm1.BtHinweiseClick(Sender: TObject);
begin
    GroupBox7.Visible:=true;
    GroupBox6.Visible:=false;
end;

end.

```