P2_A1

```c
#include <stdio.h>

int main()
{
	long long int nfak, kfak, nkfak;	// das sind jetzt 64 Bit-Variable
	int i;
	int n, k;
	int komb_6ueb4;
	int komb_8ueb6;
	long long int komb_nuebk;
	/*.....*/

	n = 13;
	k = 10;
	//n! berechnen
	nfak = 1;
	for (i = 2; i <= n; i++);
	nfak = nfak*i;

	//k! berechnen!
	kfak = 1;
	for (i = 2; i <= k; i++);
	kfak = kfak*i;

	//(n-K)! berechnen
	nkfak = 1;
	for (i = 2; i <= n - k; i++);
	nkfak = nkfak*i;

	printf("kfak %d\n", kfak);
	printf("nfak %d\n", nfak);
	printf("nkfak %d\n", nkfak);

	komb_nuebk = nfak / (kfak*nkfak);

	//printf("komb_nuebk %d\n");

	printf("nfak: %11d  kfak: %11d nkfak: %11d\n", nfak, kfak, nkfak);

	printf("%d ueber %d ergibt: %d\n", n, k, komb_nuebk); //für Ausgabe einer long long int zahl muss %11d angegeben werden

		// fuer 49 ueber 6
	komb_nuebk = 44LL * 45LL * 46LL * 47LL * 48LL * 49LL / (1 * 2 * 3 * 4 * 5 * 6);
	printf("49 ueber 6 ergibt: %11d\n", komb_nuebk);

	getchar();
	getchar();
	return 1;
}
```

P3_A1

Benutzen Sie die Vorlage spiel.c! In der Vorlage wird eine Zahl zwischen 0 und 99 pseudozufällig ermittelt. Eine weitere Zahl wird vom Benutzer des Programms eingegeben. Das Ziel ist, mit der eingegebenen Zahl die pseudozufällige Zahl zu erraten.

```
/* Programmierung 1, Praktikum 3 */
/* Vorlage                       */
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
unsigned int erzeuge_geheime_zahl(int n)
{
	time_t sekunden;
	struct tm *zeit;
	int zahl;
	time(&sekunden);
	zeit = localtime(&sekunden);

	srand(zeit->tm_sec);
	zahl = rand();
	zahl = zahl % n;
	return zahl;
}
int main(void)
{
	unsigned int geheim, geraten;
	const unsigned int N = 100;
	unsigned int anz_versuche;
	unsigned int ende = 0;
	printf("Spielen Sie mit mir!\n");

	geheim = erzeuge_geheime_zahl(N);

	//dritte Aufgabe
	anz_versuche = 0;

	//dritte Aufgabe
	while (!ende)
	{
		printf("Raten Sie eine Zahl zwischen 0 und %d:", N - 1);
		scanf("%d", &geraten);

		anz_versuche = anz_versuche + 1;

		//erste Aufgabe
		if (geraten == geheim) {
			printf("Sie haben die richtige Zahl: %d geraten", geraten);
			printf("Die zu ratende Zahl lautete %d, Sie haben %d geraten. \n", geheim,
geraten);
			ende = 1;
		}
		else {
			if (geraten < geheim)
```

```
                        {
                                printf("Sie haben die Zahl nicht erraten. Die zu ratende Zahl ist
groesser. Versuchen Sie noch mal.\n");
                        }

                        if (geraten > geheim)
                        {
                                printf("Sie haben die Zahl nicht erraten. Die zu ratende Zahl ist
kleiner. Versuchen Sie noch mal.\n");
                        }
                }
        }
        printf("Die zu ratende Zahl %d wurde nach %d Versuchen gefunden.\n", geheim,
anz_versuche);
        getchar();
        getchar();
        getchar();
        return 0;
}
```

P4_A1

Es ist ein C-Programm zu erstellen, das für eine eingegebene Jahreszahl entscheidet, ob das Jahr ein Schaltjahr ist, oder nicht.

```c
#include <stdio.h>
//Mit einer Funktion
int istSchaltjahr(int j)

{
	if (j % 4 == 0)
	{
		if (j % 100 == 0)
		{
			if (j % 400 == 0)
				return 1;
			else
				return 0;
		}
		else return 1;
	}
	else return 0;
}
int main()
{
	int j = 0;
	int schaltjahr;
	int aj, ej;
	/*  printf ("Geben Sie eine Jahreszahl ein:\n");
	scanf_s("%d", j);  */

	printf("Geben Sie das Anfangsjahr ein:\n");
	scanf_s("%d", &aj);
	printf("Geben Sie das Endjahr ein:\n");
	scanf_s("%d", &ej);
	j = aj;
	for (j = aj; j <= ej; j = j + 1) //while (j <= ej)
	{
		if (j % 4 == 0)
		{
			if (j % 100 == 0)
			{
				if (j % 400 == 0)
					printf("Schaltjahr %d:.\n", j);

				//else printf("Kein Schaltjahr.\n");
			}

			else printf("Schaltjahr: %d.\n", j);
		}
		//else printf("Kein Schaltjahr.\n");
		j = j + 1;
	}
	if (istSchaltjahr(j))
	{
		printf("Schaltjahr!\n");
	}
	else printf("Kein Schaltjahr!\n");
	getchar();
	getchar();
	return 1;
}
```

P5_A1

Schreiben Sie ein C-Programm mit einer Funktion, die von zwei double-Werten das Minimum auswählt und als Rückkehrwert bereitstellt! Testen Sie die Funktion mit zwei vorab deklarierten und initialisierten Variablen! Erweitern Sie Ihr Programm in einer geeigneten Weise, dass Sie das Minimum von drei und von vier Werten auswählen können! Dazu gibt es verschiedene Möglichkeiten.

```
#include <stdio.h>
double min(double w1, double w2)
{
if (w1 < w2) return w1;
else return w2;
}
int main()
{
double w1 = 1.20, w2 = 7.12, w3 = 0.90, w4 = 9.90;

printf("Der minimale Wert betraegt: %lf\n", min(min(w1, w2), min(w3, w4)));
getchar();
}
```

P5_A3

Es soll eine Funktion entwickelt werden, die aus einem Array vorgegebener Werte (Typ double) die folgenden Kenngrößen berechnet: Minimum Maximum Arithmetisches Mittel Für die zu berechnenden Kenngrößen sind Call-By-Reference-Parameter bereitzustellen. Benutzen Sie die als Vorlage bereitgestellte Datei arrays.c mit der Vorgabe verschiedener Arrays!

```
#include <stdio.h>
#define _USE_MATH_DEFINES
#include <math.h>

double min_(double werte[], int size)
{
int i;
double minn;
for (i = 0; i <= size - 1; i++)
{
if (werte[i] < werte[i + 1])
        minn = werte[i];
}
return minn;
}
double max_(double werte[], int size)
{
int i;
double maxx;
for (i = 0; i < size - 1; i++)
{
if (werte[i] > werte[i + 1])
maxx = werte[i];
}
return maxx;
```

```
}
double amittel(double werte[], double size)
{
int i;
double summe;
double mittelwert;
for (i = 0; i < size; i++){
summe = werte[i];
}
return summe / size;
}
int main()
{

	//int stat_auswertung(double werte[], int n, double *min, double *max, double amittel);

	int i;

	double temperaturen[24] = { 2.4, 2.0, 1.8, 1.2, 0.4, -0.6, -1.5, -1.3, 2.1, 4.7, 7.3, 8.2, 9.3, 10.2,
9.5, 7.7, 5.7, 4.6, 4.1, 3.9, 3.2, 2.9, 2.7, 2.5 };
	//int anzahl_temperaturen = 24;


	double noten[10] = { 1.7, 2.0, 2.7, 1.3, 3.0, 2.3, 1.0, 4.0, 2.3, 3.3 };
	//int anzahl_noten = 10;

	double preise[5] = { 2.50, 2.00, 3.50, 3.00, 2.20 };
	//int anzahl_preise = 5;

	printf("Minimale Note: %lf\n", min_(noten, 10));
	printf("Maximale Note: %lf\n", max_(noten, 10));
	printf("Mittelwert der Noten: %lf\n", amittel(noten, 10));
	printf("Minimale Temperatur: %lf\n", min_(temperaturen, 24));
	printf("Maximale Temperatur: %lf\n", max_(temperaturen, 24));
	printf("Mittelwert der Temperaturen: %lf\n", amittel(temperaturen, 24));
	printf("Minimaler Preis: %lf\n", min_(preise, 5));
	printf("Maximaler Preis: %lf\n", max_(preise, 5));
	printf("Mittelwert der Preisen: %lf\n", amittel(preise, 5));

	getchar();

	return 1;

}
```

P5_A2

Schreiben Sie eine Funktion, die bei einer durch Länge und Breite gegebenen rechteckigen Fläche die Anzahl der darin überlappungsfrei zu platzierenden Kreise ermittelt! Die Kreise sollen alle gleich groß sein und durch den Durchmesser beschrieben sein. Es ist nur die unten gezeigte Variante der Platzierung zu berücksichtigen. Ein Beispiel wäre eine Kiste, in die man runde Konservendosen einpackt.

```
#include <stdio.h>
#define _USE_MATH_DEFINES
#include <math.h>

int platzierung(double b, double l, double d)
{
	//b=Breite l=Laenge
	int anz_neben = b / d; //d=Durchmesser der Dosen
	int anz_unter = l / d;

	return anz_neben*anz_unter;
}
double unbenutzt(double b, double l, double d)
{
	return b*l - platzierung(b, l, d)*(d / 2)*(d / 2)*M_PI;
}
int main()
{

	int anz_neben, anz_unter;
	int anz;
	double b, l, d;
	l = 80;
	b = 40;
	d = 12;

	anz = platzierung(b, l, d);
	printf("Auf die Flaeche %2.3lfx%2.3lf passen %d Dosen mit einem %.3lf
Durchmesser.\n", b, l, anz, d);

	printf("Die unbenutzte Flaeche betraegt: %lf\n", unbenutzt(b, l, d));
	getchar();
	return 1;
}
```

P6_A1

Schreiben Sie eine C-Funktion, die ein int-Feld als Parameter übernimmt und eine Suche nach einem Wert und dessen Position im Feld vornimmt! Sollte der Wert mehrmals vorkommen, so ist die erste gefundene Position zu ermitteln. Beachten Sie, dass der gesuchte Wert möglicherweise nicht im Feld enthalten ist!

```
#include <stdio.h>
int suche_zahl(int a[], int n, int z)
{
	int i;
	for (i = 0; i < n; i++)
	{
		if (a[i] == z) return i;
	}
	return -1;  //funktioniert natuerlich nur bei positiven Werten im
Array
}
int main()
{
	int feld[10] = { 2, 4, 1, 77, 12, 45, 13, 22, 14, 19 };
	int z = 0, pos, i;
	int n = 10;  //anzahl Elemente im Feld Feldogroesse

	//1.Ausgabe
	for (i = 0; i < n; i++)
		printf("%d ", feld[i]);
	printf("\n");

	//2. Zahl einlesen
	printf("zu findende Zahl:", z);
	scanf_s("%d", &z);

	//3. Zahl im Feld suchen
	pos = suche_zahl(feld, n, z);

	//4. Ausgabe
	if (pos >= 0)
	printf("Zahl %d gefunden an Position %d.\n", z, pos);
	else printf("Zahl %d nicht gefunden.\n");

	getchar();
	getchar();
	return 1;
}
```

P6_A2

Erweitern Sie das vorgegebene C-Programm schiffe.c um das Spiel „Schiffe versenken“ spielen zu können! Dabei wird Ihnen eine pseudozufällig initialisierte quadratische Matrix seegebiet erzeugt, die Schiffe durch auf 1 gesetzte Elemente beinhaltet. Schiffe werden hier als in einer Zeile oder Spalte zusammenhängende 1-Bereiche über drei Elemente nachgebildet. Vervollständigen Sie das Programm, damit ein Spieler schrittweise die 1Positionen finden kann! Dabei sind die Versuche des Spielers zu zählen, bis er alle Schiffe „versenkt“ hat.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define N 10

void erzeuge_seegebiet(int m[][N], int kantenlaenge, int anz_schiffe)
{
	time_t sekunden;
	struct tm *zeit;
	int schiffe, z, s, r;

	time(&sekunden);
	zeit = localtime(&sekunden);
	srand(zeit->tm_sec);

	for (z = 0; z<kantenlaenge; z++)
		for (s = 0; s<kantenlaenge; s++)
			m[z][s] = 0;

	schiffe = 0;
	while (schiffe<anz_schiffe)
	{
		z = rand() % kantenlaenge;
		s = rand() % kantenlaenge;
		r = rand() % 2;   // horiz. vert.

		if (m[z][s] == 1) // bereits gesetzt
			continue;

		if (r == 0) //horizontal
		{
			if (s>0 && m[z][s - 1] == 1)
				continue;
			if (s>kantenlaenge - 3 || m[z][s + 1] == 1 || m[z][s + 2] == 1 ||
(s<kantenlaenge - 3 && m[z][s + 3] == 1))

				continue;
			if (z<kantenlaenge - 1 && (m[z + 1][s] == 1 || m[z + 1][s + 1] ==
1 || m[z + 1][s + 2] == 1))
				continue;
			if (z>0 && (m[z - 1][s] == 1 || m[z - 1][s + 1] == 1 || m[z - 1][s
+ 2] == 1))
				continue;

			m[z][s] = 1;
			m[z][s + 1] = 1;
			m[z][s + 2] = 1;
			schiffe++;
		}
		else // vertikal
		{
			if (z>0 && m[z - 1][s] == 1)
```

```
                        continue;
                    if (z>kantenlaenge - 3 || m[z + 1][s] == 1 || m[z + 2][s] == 1 ||
(z<kantenlaenge - 3 && m[z + 3][s] == 1))
                        continue;
                    if (s<kantenlaenge - 1 && (m[z][s + 1] == 1 || m[z + 1][s + 1] ==
1 || m[z + 2][s + 1] == 1))
                        continue;
                    if (s>0 && (m[z][s - 1] == 1 || m[z + 1][s - 1] == 1 || m[z + 2][s
- 1] == 1))
                        continue;
                    m[z][s] = 1;
                    m[z + 1][s] = 1;
                    m[z + 2][s] = 1;
                    schiffe++;
            }
        }
}
void anzeige_seegebiet(int m[][N], int n)

{
        int z, s;
        for (z = 0; z<n; z++)
        {
                for (s = 0; s<n; s++)
                        printf("%2d ", m[z][s]);
                printf("\n");
        }
}
int alle_schiffe_versenkt(int m[][N], int n)
{
        int z, s;
        for (z = 0; z < n; z++)
        {
                for (s = 0; s < n; s++)
                        if (m[z][s]) return 0;
//entspricht: if (m[z][s]==1), wenn wir eine 1 finden
//dann sind noch nicht alle Schiffe versenkt
        }
        return 1;
}
int main()
{
        int kl = 8; // Kantenlaenge
        int schiffe = 5; // Anzahl Schiffe
        int seegebiet[N][N];
        int z, s, v = 0;

        erzeuge_seegebiet(seegebiet, kl, schiffe);

        // dieser Funktionaufruf solle fuer das Spielen entfernt werden
        anzeige_seegebiet(seegebiet, kl);

        while (alle_schiffe_versenkt(seegebiet, kl)) {
                //entspricht: while(alle_schiffe_versenkt(seegebiet, kl)==0)
wenn noch nicht alles versenkt

                // ab hier Spiel programmieren
                //1. Zeile und Spalte fuer Tipp eingeben
                int z, s;
```

```
            printf("Eingabe Zeile: "); scanf_s("%d", &z);
            printf("Eingabe Spalte: "); scanf_s("%d", &s);

            //2. Test ob eingegebene Werte in Matrix liegen
            if (z < 0 || z >= kl || s < 0 || s >= kl) {

                  //Idex ausserhalb der Matrix
                  printf("Schluss ausserhalb Bereich");
            }
            else if (seegebiet[z][s])
            { //Wir haben ein Schiff getroffen
                  printf("Treffer!\n");
                  seegebiet[z][s] = 0; //getroffene Stelle/Schiff markieren
mit 0, da im naechsten zu weg

            }
            else
                  printf("Schuss ins Wasser!\n"); //4. - das war nichts
            v++;
      }
      getchar();
      getchar();
      return 1;
}
```

P7_A1

Aufgabe 1: Binäre Suche Schreiben Sie eine C-Funktion, die ein aufsteigend sortiertes int-Feld als Parameter übernimmt und eine Suche nach einem Wert und dessen Position im Feld vornimmt! Beachten Sie, dass der gesuchte Wert möglicherweise nicht im Feld enthalten ist!

```
/*Zur Demonstration der Arbeitsweise von suche_binaer() lassen wir uns li, re, mitte und feld[mitte] ausgeben.
Die Ausgaben sind aber nicht gefordert.Typischerweise sollte eine Funktion nichts per printf ausgeben
*/


#include <stdio.h>

int suche_einfach(int feld[], int n, int s)
{
int i;
for (i = 0; i < n; i++) {

if (feld[i] == s) return i;
} return -1;
}


int suche_binaer(int feld[], int n, int s)
{

int li, re, mitte = 0;
li = 0;
re = n - 1;

while (li<re)  //solange Bereich noch mehr als ein Element umfasst

{

mitte = (li + re) / 2;

printf("li=%02d (Wert[mitte]:%02d) re=%02d \n", li, mitte, feld[mitte], re);
//Wert in der Mitte mit dem gesuchten Wert vergleichen
if (feld[mitte] == s)
{
return mitte;
}
else if (feld[mitte] < s)
{
//rechts weitersuchen
li = mitte + 1;
}
else //feld[mitte]>s
{
//links weitersuchen
re = mitte - 1;
}
}

// wenn Bereich auf ein Element geschrumpft, dann direkt Element  vergleichen
printf("Bereich endgueltig eingeschraenkt auf Indexpos: %02d (Wert[%02d]: %02d)\n", li, li, feld[li]);
if (feld[li] == s)
```

```
return li;
else
return -1;
}
int main()
{
int werte[10] = { 3, 5, 7, 8, 12, 18, 22, 34, 36, 45 };
int n = 10;
int suchwert=0, pos;

suchwert = 12;

//pos= suche_einfach(werte, n, suchwert);  //gehört zu der ersten (einfachen) Funktion

pos = suche_binaer(werte, n, suchwert);
if (pos >= 0)
printf("Wert: %d gefunden an Position %d\n", suchwert, pos);
else
printf("wert: %d nicht gefunden.\n", suchwert);


getchar();
getchar();
return 1;
}
```

P7_A2

a) Geben Sie die Adressen von preis1, preis2 und von den ersten drei Elementen des Feldes werte aus?
b) Tauschen Sie die Werte von preis1 und preis2 mit einer der beiden Funktionen sw1() und sw2()! Welche Funktion kann für den Tausch verwendet werden? Warum?

```
#include <stdio.h>
void sw1(double a, double b)
{
   double t;
   t = a;  a = b;  b = t;
}
void sw2(double *a, double *b)
{
   double t;
   t = *a;  *a = *b;  *b = t;
}
int main()
{
   int n = 10;
   double preis1 = 22.99, preis2 = 21.49;
   double werte[10] = { 9, 8, 7, 6, 5, 4, 3, 2, 1, 0 };
   double *d_zeiger;
   printf("Der Wert von n ist %d, die Adresse von n ist %x \n", n, &n);
   // hier erweitern
   printf("Der Wert von preis1 ist %.2lf, die Adresse von preis1 ist %x \n", preis1, &preis1);
   printf("Der Wert von preis2 ist %.2lf, die Adresse von preis2 ist %x \n", preis2, &preis2);

   //int e1=0;
   //e1=werte[0];
   int i;
   for (i = 0; i < 3; i++)
   {
      printf("Der Wert von dem %d.Element ist %0.lf, die Adresse von dem ersten Element ist %x \n",
i, werte[i], &werte[i]);
   }
   printf("Wir studieren die Auswikrung der Funktionen sw1() und sw2()\n");

   printf("Vorher: preis1=%lf, preis2=%lf\n", preis1, preis2);

   sw1(preis1, preis2);

   printf("Nach Aufruf sw1(): preis1=%lf, preis2=%lf\n", preis1, preis2);

   sw2(&preis1, &preis2);  //hier muss man die Variablen als Adresse übergeben

   printf("Nach Aufruf sw2(): preis1=%lf, preis2=%lf\n", preis1, preis2);

   getchar();
   return 0;
}
```

P7_A3

Schreiben Sie verschiedene Funktionen, die eine 8x8 Matrix aus int-Elementen als Parameter übernehmen und wie folgt arbeiten:

a) Formatierte Ausgabe der Matrix. Jedes Element soll dezimaler Zahlenwert mit Platz für 10 Stellen ausgegeben werden (per "%10d" bei printf()). b) Schachbrettartige Belegung der Matrix mit 0- und 1-Werten. Es soll mit dem durch 0,0 indizierten Element mit dem Wert 1 begonnen werden. c) die ersten 40 Matrixelemente mit den Fibonacci-Zahlen belegt: Jedes Matrixelement mit den Indizes z,s (Zeile, Spalte) soll mit der Zahl fibo(z*8+s) belegt werden. Dabei werden die Zeilen und Spalten im Bereich von 0 bis 7 indiziert.

```c
//Praktikum 7 Aufgabe 3
#include <stdio.h>
#define N 8
int fibo(int n)
{
	if (n < 2) return 1;
	else return fibo(n - 2) + fibo(n - 1);
}
void matrix_setNull(int ma[][N])
{
	int z, s;
	for (z = 0; z < N; z++) {
		for (s = 0; s < N; s++) {
			ma[z][s] = 0;
		}
	}
}
void matrix_ausgabe(int ma[][N])
{
	int z, s;
	for (z = 0; z < N; z++) {
		for (s = 0; s < N; s++) {
			printf("%10d", ma[z][s]);
		}
	}
}
void matrix_schachbrett(int ma[][N])
{
	int z, s;
	int wert;
	for (z = 0; z < N; z++) {  //gerade
		if (z % 2 == 0)  wert = 1;
		else wert = 0;
		for (s = 0; s < N; s++) {
			ma[z][s] = wert;
			if (wert == 1)  wert = 0;
			else wert = 1;
		}
	}
}
void matrix_fibo(int ma[][N])
{
	int z, s;
	int zaehler = 0;
	for (z = 0; z < N; z++) {
		for (s = 0; s < N; s++) {
			if (zaehler < 40) {
				ma[z][s] = fibo(zaehler);
				zaehler++;
			}
			else ma[z][s] = 0;
		}
```

```
        }
}
int main()
{
        //int z, s;
        int matrix[N][N];

        matrix_setNull(matrix);
        printf("\nDie Matrix nach dem Nullsetzen aller Elemente:\n");
        matrix_ausgabe(matrix);

        matrix_schachbrett(matrix);
        printf("Die Matrix nach dem Belegen mit 0, 1 (schabrettartig):\n");
        matrix_ausgabe(matrix);

        matrix_fibo(matrix);
        printf("Die Matrix nach mit Fibonacci-Werten:\n");
        matrix_ausgabe(matrix);
        matrix_ausgabe(matrix);

        getchar();
        return 1;

}
```

P8_A1

Für einen ganzzahligen, positiven Exponenten n und einen reellen Wert x gilt: = 1

x^n = 1, wenn n=0
x^n= (x^(n/2))^2 wenn n geradzahlig
x^n=x(x^((n-1)/2)^2 wenn n ungerade*

a) Berechnen Sie damit 217. Zählen Sie dabei die Anzahl der Multiplikationen!
b) Schreiben Sie eine rekursive Funktion, die xn nach dem oben angegebenem Prinzip berechnet!
c) Begründen Sie, warum dies effizienter ist als ein naiver Algorithmus,der xn durch fortwährendes Multiplizieren mit x berechnet!

```c
#include <stdio.h>
double potenz(double x, int n)
{
	double w; //zwischenergebnis
	if (n == 0)
	return 1.0;

	if (n % 2 == 0)//geradzahliger Exponent
	{
		//potenz aufrufen und noch etwas rechnen
		w = potenz(x, n / 2);
		w = w*w;
		return w;
	}
	else //ungerader Exponent
	{
		//auch hier potenz aufrufen und noch etwas rechnen
		w = potenz(x, (n - 1) / 2);
		w = w*w;
		w = x*w;
		return w;
	}
}
int main()
{
	double erg = 0, x;
	int n;
	x = 2;
	n = 17;
	erg = potenz(x, n);

	printf("%lf hoch %d ergibt %lf\n", x, n, erg);

	getchar();
	return 1;

}
```

P8_A2

In dem C-Programm texte.c sind verschiedene Zeichenketten vorgegeben, die als Aussagen zum Teil richtig, aber auch falsch sein können. Die Zeichenketten sind zeilenweise in einer Matrix abgelegt.
Teilaufgaben:
a) Geben Sie alle Zeichenketten nacheinander aus!
b) Erweitern Sie das Programm um eine manuelle Bewertung (richtig oder falsch) per Konsoleneingabe! Speichern Sie sich die Entscheidung programmintern in einer geeigneten Weise ab!
c) Geben Sie nach der Bewertung alle richtigen Aussagen noch einmal aus!
d) Geben Sie danach alle falschen Aussagen noch einmal aus!
e) Lassen Sie die Anzahl der richtigen und die Anzahl der falschen Aussagen ausgeben!

```
#include <stdio.h>
#define TEXT_LEN 160
#define N 10

int main()
{
	char alle_aussagen[N][TEXT_LEN] = {
		"Hunde fressen Voegel.",
		"Die Elbe fliesst zur Nordsee.",
		"Informatik gab es bereits in der Steinzeit.",
		"Zuviel Alkohol ist ungesund.",
		"Autofahren verbessert die Luftqualitaet in der Stadt.",
		"Kaffee macht munter.",
		"Jede Funktion hat eine Nullstelle.",
		"Die Erde ist eine Scheibe.",
		"Studieren ist freiwillig.",
		"Es gibt den Weihnachtsmann." };

	// a)Ausgabe aller Zeichenketten
	typedef enum { richtig, falsch } entsch_t;
		entsch_t e[N];

		int i;
		for (i = 0; i < N; i++) {
			printf("  %s \n", alle_aussagen[i]);
		}

		// b) Entscheiden ob wahr oder falsch
	int  n_richtige, n_falsche;
	char antw;
	for (i = 0; i < N; i++) {
		printf("%s\n", alle_aussagen[i]);
		while (1)
		{
			printf("Ist das richtig[j/n]?");
			scanf_s("%c", &antw);
			if (antw == 'j' || antw == 'J') {
				e[i] = richtig; //e ist ein Feld
				break;
			}
			if (antw == 'n' || antw == 'N') {
				e[i] = falsch;
				break;
```

```
                        }
                }
        }
        // c) Alle richtigen nochmal ausgeben
        printf("Richtige Aussagen:\n");
        for (i = 0; i < N; i++) {
                if (e[i] == richtig) {
                        printf("%s\n", alle_aussagen[i]);
                }
        }
                // d) Alle falschen nochmal ausgeben
                printf("Falsche Aussagen: ");
                for (i = 0; i < N; i++) {
                        if (e[i] == falsch) {
                                printf("%s\n", alle_aussagen[i]);
                        }
                }
                // e) Zaehlen der richtigen und falschen
                n_richtige = 0;
                n_falsche = 0;
                for (i = 0; i < N; i++) {
                        if (e[i] == richtig)
                                n_richtige++;
                        if (e[i] == falsch)
                                n_falsche++;
                }

                printf("Anzahl der richtigen Aussagen: %d\n Anzahl der falschen Aussagen: %d",
n_richtige, n_falsche);

        getchar();
        getchar();
        return 1;
}
```