Mid/20	A1/30	A2/12	A3/8	A4/10	A5/10	A6/10	SUM/100	Note

Aufgabe 1 (30 Punkte)

Jede richtig beantwortete Frage ergibt **2 Punkte**. Jede falsch-, mehrfach- oder nicht beantwortete Frage ergibt **0 Punkte**. Je Frage ist nur eine Antwort richtig.

1.1 Wenn der Vektorname `arr` ein Zeiger auf das erste Element `arr[0]` ist, zeigen folgende Anweisungen auf das zweite Element des Arrays:

A. `&arr[2], arr+2`
B. `&arr[1], arr+1`
C. `*arr+2, arr[2]`
D. `arr+2, &arr[1]`

1.2 Was liefert das Programm `"prog"`, wenn alle Funktionen verfügbar sind und es wie folgt, auf der Konsole ausgeführt wird:

```
C:\ prog 4 9
//prog.c
int main(int argc, char *argv[]){
    printf("%.2f \n", sqrt(atof(*(argv+1))*atof(*(argv+2))));
    return 0;
}
```

A. `36.00`
B. `6.000000`
C. `6.00`
D. `3.60`

1.3 Der Wert von x in dezimaler Notation beträgt:

```
int x = (1 << 7) ^ (255 >> 7);
```

A. `1`
B. `129`
C. `128`
D. `255`

1.4 Wenn der Zeiger `ptr` auf eine Struktur vom Typ `struct address` zeigt, repräsentiert der folgende Ausdruck das Element `plz` des Objekts:

A. `(*address).plz`
B. `*adress.plz`
C. `ptr->*plz`
D. `ptr->plz`

1.5 Die Ausgabe des nachfolgenden Codes lautet:

```
#define FUNC(x,y )(x*y)/2
int a = 3;
printf("%d ", FUNC(a+5, a-2));
```

A. `8`
B. `4`
C. `5.5`
D. `6`

1.6 Welchen Speicherplatz benötigt die Variable `pc1 mindestens`?

```
union ipAdd {struct{unsigned char oct1, oct2, oct3, oct4;}eth0 , eth1,
eth2, wlan0, wlan1;};
union ipAdd pc1;
```

A. 1 Byte
B. 4 Byte
C. 5 Byte
D. 20 Byte

1.7 Geben Sie eine Möglichkeit zur Initialisierung des Elementes oct1 des Union-Elementes eth2 der Variable pc1 in der Frage 1.6 mit dem Wert 10.

A. `pc1.oct1.eth2 = "10";`
B. `pc1.oct2.eth2 = 10;`
C. `pc1.eth1.oct1 = 10;`
D. `pc1->eth2->oct1 = 10`

1.8 Welchen hexadezimalen Wert hat x nach der Ausführung des Ausdrucks?

```
int x = 0x25 ^ (031 / 3);
```

A. 10
B. 14
C. 45
D. 2d

1.9 Definieren Sie einen Zeiger auf eine Funktion `func` mit dem Rückgabe-Typ Zeiger auf `long` und Übergabeparameter Zeiger auf `int`.

A. `long* (*func)(int *);`
B. `int* (*func)(long *);`
C. `long* *func(int *);`
D. `long (*func)(int *);`

1.10 Welchen Wert hat der Ausdruck `p-v` mit folgenden Definitionen?

```
double v[] = {5, 10, 15, 0}; double *p = &v[3];
```

A. 2
B. 24
C. 3
D. -3

1.11 Wie lautet die Ausgabe des folgenden Codes? Es handelt sich dabei um eine 32-Bit-Architektur.

```
struct node{int data; struct node *link;} *p;
printf("%d, %d\n", sizeof(*p), sizeof(p));
```

A. 4, 4
B. 8, 8
C. 4, 8
D. 8, 4

1.12 Wie kann ein reservierter Speicherbereich frei gegeben werden?

A. `remove(variable)`
B. `delete(variable)`
C. `calloc(variable)`
D. `free(variable)`

1.13 Nach der Ausführung des folgenden Codes ist der Wert von `var1->a`:

```
union var {int a, b;};
union var *var1=malloc(sizeof(union var));
var1->a=20; (*var1).b=30;
```

A. 10
B. 20
C. 30
D. 40

1.14 Die int-Zeiger p1 und p2 zeigen auf die int-Variablen a und b. Dann wird durch folgende Anweisung:

```
printf("%d\n", *(p1 + p2));
```

A. Zur Laufzeit ein undefinierter Wert angezeigt.
B. Der Inhalt der Adresse p1+p2 angezeigt.
C. Der Compiler eine Fehlermeldung ausgeben.
D. der Wert der Summe *p1+*p2, also a+b angezeigt.

1.15 In welcher Reihenfolge werden die Werkzeuge in der Entwicklung mit C typisch eingesetzt?

A. Präprozessor -> Compiler -> Linker
B. Compiler -> Präprozessor -> Linker
C. Präprozessor -> Compiler -> Locator
D. Compiler -> Locator -> Linker

Aufgabe 2 (12 Punkte):

2.1 Schreiben Sie eine C-Funktion, die eine Ganzzahl, einen Zeiger auf ein sortiertes Array mit Ganzzahlen und die Länge des Arrays bekommt. Die Funktion soll die Stelle der Ganzzahl im sortierten Array nach dem Algorithmus der binären Suche finden und zurückgeben. Bei nichterfolgreicher Suche soll eine Meldung zurückgegeben werden, die in der aufrufenden Funktion entsprechend interpretiert werden kann.

2.2 Schreiben Sie eine Funktion `main`, in der Sie ein Array deklarieren und mit sortierten Ganzzahlen initialisieren, die Funktion in 2.1 aufrufen und das Ergebnis ausgeben.

Hinweis: Die binäre Suche erfolgt nach dem "Teile und Herrsche" Prinzip. Zuerst wird das gesuchte Element mit dem Element in der Mitte des sortierten Arrays verglichen. Ist das gesuchte Element kleiner, wird in der unteren Hälfte des Arrays weitergesucht. Ist es hingegen größer, wird die obere Hälfte gesucht.
In der zu untersuchenden Hälfte (und erneut in den folgenden Hälften) wird genauso verfahren: Das mittlere Element liefert wieder die Entscheidung darüber, ob und wo weitergesucht werden muss.

Aufgabe 3 (8 Punkte):

3.1 Implementieren Sie eine rekursive Multiplikationsfunktion die ohne den Operator '*' auskommt.

```
int mult(int a, int b)
```

```
Bsp.:
```

```
 3 * 5 == 5 + (2 * 5) == 5 + (5 + (1 * 5)) == 5 + 5 + 5
```

3.2 Wieviele Rekursionsschritte erzeugt der Aufruf `mult(2, 5)`? Wie viele Rekursionsschritte erzeugt der Aufrufe `mult(5, 2)`?

Aufgabe 4 (10 Punkte):

Auf der nächsten Seite finden Sie den Quelltext des Programms `incorrect`, das die Zahlen in Packet sortieren soll. Das Programm wird wie folgt erzeugt:

```
gcc –Wall incorrect.c –o incorrect
```

Aufgabe:

Der Quelltext enthält zehn syntaktische Fehler, die von Ihnen korrigiert werden sollen. Die Korrektur soll durch Angaben nach folgendem Schema auf dem beiliegenden Lösungsblatt erfolgen:

< Zeilennummer>: vollständige, korrigierte Quelltextzeile

```
1   /* Filename: incorrect.c
2
3   #define NumOfPackets = 5
4   void MySort(double * , int , int );
5   #include <stdio.h>
6
7   int main (){
8         int iCnt;
9         static double Packet[NumOfPackets]={3.01,2.08,5.02,8.52,7.10};
10
11        MySort(Packet, 0, NumOfPackets-1);
12
13        for (iCnt=0; iCnt < NumOfPackets; iCnt++)
14        {
15              printf("Value %d is %lf\n", iCnt+1, Packet[iCnt]) + 1;
16        }
17        return(0)++;
18  }
19
20  void MySort(double pArrToSort[&], int iStartIdx, int iEndIdx)
21  {
22       int iUp= iStartIdx,iDown= iEndIdx;
23       double dMedVal, dTmp="Y";
24
25       dMedVal = pArrToSort[(iStartIdx+iEndIdx) / 2];
26
27       do
28       {
29            while(pArrToSort[iUp] < dMedVal)
30                 iUp++;
31            while(pArrToSort[iDown] > dMedVal)
32                 iDown--
33
34              if( iUp <= iDown)
35              {
36                    dTmp = pArrToSort[iUp];
37                    pArrToSort[iUp] = pArrToSort[iDown];
38                    pArrToSort[iDown] = dTmp;
39                    iUp++;
40                    iDown--;
41              }
42       }
43       while(iUp < iDown)
44
45       if(iStartIdx < iDown)
46       {
47              MySort(pArrToSort,iStartIdx,iDown);
48       }
49       if(iUp < iEndIdx)
50       {
51              &MySort(pArrToSort,iUp,iEndIdx);
52              }
53        }
54  }
```

Aufgabe 5 (10 Punkte):

Punkte auf der Oberfläche der Erde lassen sich durch die Angabe von Längen- und Breitengrad festlegen. Der Wert für den Längen- bzw. Breitengrad wird im Sexagesimalformat angegeben. 1 Grad ist dabei unterteilt in 60 Minuten, 1 Minute ist unterteilt in 60 Sekunden. Ein Beispiel für eine Angabe im Sexagesimalformat ist 50 Grad, 30 Minuten, 15,0 Sekunden (50°30'15,0''). Die Darstellung dieses Wertes in Grad mit Dezimalstellen ist 50 + 30 / 60 + 15,0 / 3600 = 50,5042 Grad. Gehen Sie für diese Aufgabe davon aus, dass alle Werte positiv (>0) sind.

5.1 a. Entwerfen Sie einen C-Datentyp, der Werte im Sexagesimalformat aufnehmen kann. Sehen Sie für Grad und Minuten einen Ganzzahltyp vor, die Sekunden sollen als Fließkommazahl gespeichert werden.

b. Entwerfen Sie einen C-Datentyp zur Darstellung eines Punktes auf der Erdoberfläche. Ein Punkt soll durch Längen- und Breitengrad im Sexagesimalformat beschrieben werden.

5.2 Implementieren Sie eine Funktion, die als Eingabe einen Wert im Sexagesimalformat bekommt und diesen in Grad mit Dezimalstellen umrechnet und zurückgibt.

5.3 Implementieren Sie eine Funktion, die als Eingabe einen Wert in Grad bekommt und diesen in Radiant umrechnet und zurückgibt. Hinweis: `1° = 3,14 / 180° rad`.

Auf der Erde lässt sich der Abstand zweier Punkte näherungsweise mit folgender Formel berechnen:

```
lat = (lat1 + lat2) / 2
dx = 111,3 * cos(lat) * (lon1 - lon2)
dy = 111,3 * (lat1 - lat2)
distance = sqrt(dx * dx + dy * dy)
```

Dabei sind `(lon1, lat1)` der Längen- und Breitengrad des ersten Punktes, `(lon2, lat2)` beschreiben den zweiten Punkt. `lat`, `dx`, `dy` und `distance` sind Hilfsvariablen.

5.4 Implementieren Sie eine Funktion, die zwei Punkte auf der Erdoberfläche als Eingabe bekommt und mit der Näherungsformel den Abstand der Punkte berechnet und diesen zurückgibt.

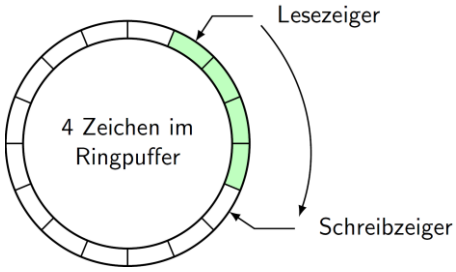
Hinweis: Die cos-Funktion erwartet den Eingabeparameter in Radiant.

5.5 Komplettieren Sie das C-Programm: Implementieren Sie die main-Funktion. Legen Sie zwei Variablen zur Aufnahme je eines Punktes auf der Erdoberfläche an und initialisieren Sie diese mit Phantasiekoordinaten. Lassen Sie die Distanz zwischen den Punkten berechnen und geben Sie das Ergebnis auf dem Bildschirm aus. Ergänzen Sie evtl. benötigte Header-Dateien.

Aufgabe 6 (10 Punkte):

Ringpuffer sind nützliche Datenstrukturen in viele Zweigen der Technik, z.B. beim temporären Schreiben von Log-Daten oder auch als Tastaturpuffer.

Ein Ringpuffer besteht aus einer zirkulären Datenstruktur fester Größe, z.B. ein Array. Weiterhin gibt es einen Schreib- und einen Lesezeiger (das müssen nicht unbedingt Zeiger/Pointer in C sein). Die Zeiger bezeichnen das Element, auf dem die nächste Schreib- bzw. Leseoperation stattfinden wird. Nach der Operation wird der Zeiger entsprechend weitergesetzt. Zirkulär bedeutet in diesem Fall, dass nach einer Schreiboperation auf das letzte Element die nächste Schreiboperation auf dem nullten Element des Arrays stattfinden wird. Gleiches gilt für die Leseoperation.



In dieser Aufgabe soll ein Ringpuffer zur Speicherung von ASCII-Zeichen implementiert werden.

6.1 Erstellen Sie einen Datentyp `buf_t`, der einen Ringpuffer repräsentieren soll. Der Datentyp soll folgende Eigenschaften speichern können:

a. Den eigentlichen Puffer, das Array. Typ: `char *`
b. Die Größe des Puffers in Elementen. Typ: `int`
c. Einen Lesezeiger. Typ: `int`
d. Einen Schreibzeiger. Typ: `int`

6.2 Implementieren Sie die folgenden drei Funktionen:

a. Erzeugen eines Ringpuffers

Die Funktion bekommt als Eingabe einen Zeiger auf eine Datenstruktur, die den Ringpuffer repräsentiert und die gewünschte Größe des Puffers. Die Funktion alloziert Speicher für den Puffer, initialisiert den Speicher mit Leerzeichen und setzt die anderen Eigenschaften, z.B. die Größe, passend.

```
void create(buf_t *buf, int size);
```

b. Schreiben

Die Funktion bekommt als Eingabe einen Zeiger auf einen bestehenden Ringpuffer und das Zeichen das in den Puffer geschrieben werden soll.

```
void write(buf_t *buf, char ch);
```

c. Lesen

Die Funktion bekommt als Eingabe einen Zeiger auf einen bestehenden Ringpuffer und einen Zeiger auf `char`, um das gelesene Zeichen an den Aufrufer zurückzugeben.

```
void read(buf_t *buf, char *ch);
```

6.3 Implementieren Sie die main-Funktion. Lesen Sie die gewünschte Größe des Ringpuffers aus dem Kommandozeilenargument. Hat der Benutzer keine Kommandozeilenargumente angegeben, nutzen Sie eine feste Größe. Legen Sie einen Ringpuffer der gewünschten Größe an. Rufen Sie die read-Funktion auf. Welches Zeichen bekommen Sie zurück?