Musterlösung: DIV (Magister) 1. Serie

1.a) geg. Laufschrift (endlich), Array der Buchstaben, Array der Anzahlen

- gehe durch die Laufschrift von vorn nach hinten (z.B. mit for(i=0;i<L.length;i++))
- dabei: arbeite jeden einzelnen Buchstaben ab: (eventuelle Leerzeichen und Satzzeichen werden ignoriert); vergleiche Buchstabe mit Array, Buchstabe neu ? – erzeuge Arrayeintrag mit 1
  Buchstabe vorhanden ? – erhöhe dazugehörigen Arrayeintrag um 1
- Ende: durchlaufe Array -> merke immer höchsten Wert -> gib am Ende den korrespondierenden Buchstaben des höchsten Arraywertes aus

2 Punkte

1.b) geg. Buch (endlich), Liste/Array der Wörter, Array der Anzahlen

- gehe durch das Buch von Anfang bis zum Schluß
- dabei: lies und speichere jeden Buchstaben -> bilde einen String
  bis man auf ein Leerzeichen oder ein Satzzeichen stößt:
  vergleiche die soeben gelesene Buchstabenkette (String) mit den schon abgespeicherten Buchstabenketten

-> Neu ? : speichern & Wert 1 im Anzahlarray : erhöhe zugehörigen Wert um 1 im Anzahlarray

- Ende: durchlaufe Array -> merke immer höchsten Wert -> gib am Ende das korrespondierende Wort des höchsten Arraywertes aus

2 Punkte

1.c) periodische Laufschrift -> je nach Interpretation endlich oder unendlich

endlich: wie (a), Algorithmus muss aber nur für eine Periode ausgeführt werden, Periodenlänge kann erkannt werden...

unendlich: Algorithmus aus (a) würde nie terminieren, die Periodenlänge kann nie sicher erkannt werden...

2 Punkte

Jeder Schritt definiert ?
Festgelegte Folge von Ausführungsschritten ?

2.) Restberechnung nur mit +, -, * (mehrere Möglichkeiten)

```
int modulo(int x, int y)                //1 Punkt für Ein- oder Übergabe
//für positive Eingaben
{
        if (y==0) return (x);           //1 Punkt für Abfangen Fall y==0
        while(x>=y)
        {
                x-=y;                   //3 Punkte für While-Schleife
        }
        printf("Ergebnis: %d", x);      //1 Punkt für Aus- oder Rückgabe
        return (x);
}
```

3.)

```
#include <stdio.h>

void ggT_euklid(int m, int n) {
        int ggT,tmp,rest;

        do {
          if(n < m) {     // tauschen
                tmp = m;
                m = n;
                n = tmp;
          };
          rest = n % m;   // größer mod kleiner
          tmp  = m;
          n = m;
          m = rest;
        } while(rest);
        ggT = tmp;          // kleinere Zahl ausgeben
        printf("\nggT: %i\n",ggT);
}

int main()
 {
        ggT_euklid(35,20);          // Beispielwerte
        return 0;
}
```

4 Punkte

4.) **Algo findet <u>größte Integer-Zahl</u> (auch negativ) im <u>Array von 0 bis N-1</u>**
<u>Kommentare (</u>4 Punkte)

```
#include <stdio.h>
int max(int* a,int N) {     // Pointer des Arrays und Länge als Parameter
        int tmp=a[0],i;     // erster Arrayeintrag ausgelesen

        for(i=1 ; i < N; i++)         // "eins daneben" Prinzip
                if(tmp < a[i]) tmp = a[i];
        return tmp;
}
int main() {
        int i,a[10]; // N=10 als Beispiel
        for(i=0; i<10 ; i++) a[i] = (i * 17) % 15;       // Init mit Werten zwischen 0 und 14
        printf("\nMax im Array: %i\n",max(a,10));
        return 0;
}
```

5.) Beweise mit Induktion:

a) S(n): $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

Induktionsanfang:

Für S(1):

$$\sum_{i=1}^{1} i = \frac{n(n+1)}{2}$$

$$1 = \frac{1(1+1)}{2}$$

$1 = 1$ w. A.

Induktionsvoraussetzung: S(n)

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

Induktionsbeweis: S(n+1) unter Annahme der Gültigkeit von S(n)

$$\text{S(n+1)}: \sum_{i=1}^{n+1} i = \frac{(n+1)((n+1)+1)}{2} = \frac{(n+1)(n+2)}{2}$$

$$\sum_{i=1}^{n+1} i = \sum_{i=1}^{n} i + (n+1) = \text{S(n)} + (n+1)$$

$$= \frac{n(n+1)}{2} + \frac{2(n+1)}{2} \qquad \text{//(n+1) ausklammern}$$

$$= \frac{(n+1)(n+2)}{2}$$

$$= S(n+1)$$

b)

$$\text{S(n)}: \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

Induktionsanfang: S(1)

$$\sum_{i=1}^{1} i^2 = \frac{n(n+1)(2n+1)}{6}$$

$$1 = \frac{1(1+1)(2*1+1)}{6}$$

$1 = 1$ *w. A.*

Induktionsvoraussetzung: S(n)

$$\text{S(n)}: \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

Induktionsbeweis: S(n+1) unter Annahme der Gültigkeit von S(n)

$$\text{S(n+1)}: \sum_{i=1}^{n+1} i^2 = \frac{(n+1)((n+1)+1)(2(n+1)+1)}{6} = \frac{(n+1)(n+2)(2n+3)}{6}$$

Schreibe

$$\sum_{i=1}^{n+1} i^2 = \sum_{i=1}^{n} i^2 + (n+1)^2 = S(n) + (n+1)^2$$

$$= \frac{n(n+1)(2n+1)}{6} + \frac{6(n+1)^2}{6}$$

$$= \frac{(n+1)\,(2n^2+7n+6)}{6}$$

$$= \frac{(n+1)(n+2)(2n+3)}{6}$$

und das ist gerade S(n+1) wie oben berechnet.