Aufgabe 1: Aufwandsabschätzung

a) Sei n die Länge des Arrays

```
static int DoubleSum(int[] F){
    int cnt = 0;
    For(int i = 1; i < F.Length - 1; i += 2){        | n/2 mal.
        int sum = 0;                                 | 1
        for(int j = i-1; j < i+2; j++){              | 3 mal.
            sum += F[j];                             | 2.
            cnt++;                                   | 2.
        }                                            | 6.
        if(sum > 0)                                  | 2.
            F[i] = F[i] | sum;                       | 2.
    }                                                | 9.
    return cnt;                                      | 1
}
```

$\log \frac{n}{2} + 2$

$f(n) = \frac{9}{2}n + 2$

Sei $c = 5$ und $g(n) = n$

Sei $n_0 \in \mathbb{N}$: $\frac{9}{2} n_0 + 2 \leq 5n$

$\frac{1}{2} n_0 \geq 2$

$n_0 \geq 4$

Also ab $n_0 = 4$ $\quad f(n) \leq c \cdot g(n)$

also $\boxed{f(n) = O(n)}$

b) static int DownWas2(int[] F){

```
int Cnt = 0;
for(int i = 1; i < F.Length; j++){
    int sum = 0;
    for(int j = i+1; j < F.Length; j++){
        Sum += F[j];
        Cnt++;
    }
    if(sum > 0)
        F[i] = F[i] / sum;
}
return Cnt;
}
```

$$
\begin{array}{l}
1 \\
+ \\
(n-1 - mal) \\
\quad n-1 \cdot mal \\
\quad + \\
\quad 2 \cdot \sum_{i=1}^{n-1} (n-1-i) \\
+ \\
2 \cdot (n-1) \\
+ \\
1
\end{array}
$$

$$f(n) = 1 + (n-1) + 2 \cdot \sum_{i=1}^{n-1} (n-1-i) + 2(n-1) + 1$$

$$= (n-1) + 2 + 2(n-1) + 2\left[(n-1)(n-1) - \frac{(n-1)(n-1+1)}{2}\right]$$

$$= (n-1) + 2 + 2(n-1) + 2\left[(n-1)\left[n-1 - \frac{n}{2}\right]\right]$$

$$= (n-1) + 2 + 2(n-1) + 2\left[(n-1)\left[\frac{n-2}{2}\right]\right]$$

$$= (n-1) + 2 + 2(n-1) + (n-1)(n-2)$$

$$= (n-1) + 2 + 2(n-1) + n^2 - 3n + 2$$

$$= n^2 + 1.$$

sei $c = 2$ und $g(n) = n^2$ und $n_0 \in \mathbb{N}$

ab welchem n_0 ist $f(n_0) \leq c \cdot g(n)$

$n^2 + 1 \leq 2n^2$

$n^2 \geq 1$

also ab $n_0 = 1$ $f(n) \leq c \cdot g(n)$ also $f(n) = O(n^2)$

Aufgabe 2: Countingsort

a) Zu sortieren: 4 3 0 1 4 2 3 7 3

4	3	0	1	4	2	3	7	3

0	1	2	3	4	5	6	7
1	1	1	3	2	0	0	1

B:

0	1	2	3	4	5	6	7
1	2	3	6	8	8	8	9

C:

1	2	3	4	5	6	7	8	9
					3			

B:

0	1	2	3	4	5	6	7
1	2	3	5	8	8	8	9

C:

1	2	3	4	5	6	7	8	9
					3			7

B:

0	1	2	3	4	5	6	7
1	2	3	5	8	8	8	8

C:

1	2	3	4	5	6	7	8	9
				3	3			7

B:

0	1	2	3	4	5	6	7
1	2	3	4	8	8	8	8

C:

1	2	3	4	5	6	7	8	9
				3	3			7

B:

0	1	2	3	4	5	6	7
1	2	2	4	8	8	8	8

C:

1	2	3	4	5	6	7	8	9
		2		3	3		4	7

B:

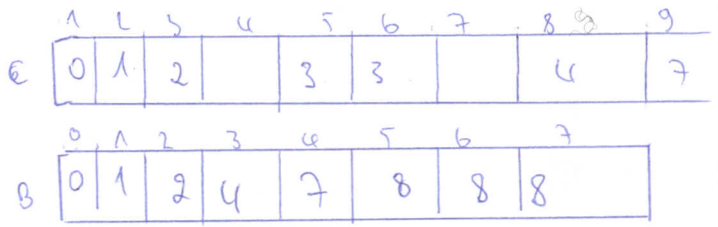
0	1	2	3	4	5	6	7
1	2	2	4	7	8	8	8

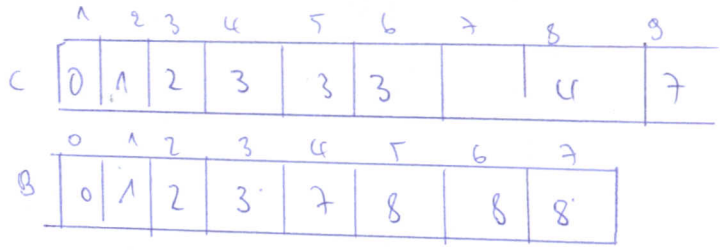
C:

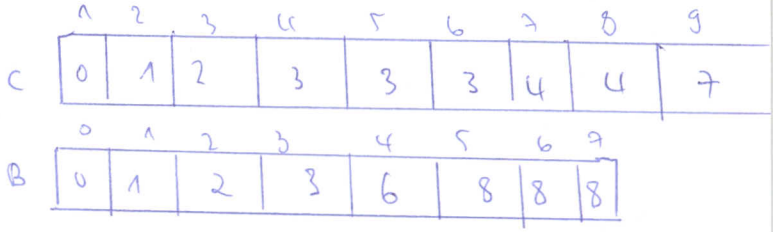
1	2	3	4	5	6	7	8	9
	1	2		3	3		4	7

B:

0	1	2	3	4	5	6	7
1	1	2	4	7	8	8	8







2) Wenn man das Eingabearray von vorn nach hinten ist einfügt, ist das Verfahren nicht stabil, da das Eingabe Array die Element vom 1 bis n (anzahl der Element) enthält.

Aufgabe 3: Hashing zum Ersten

a) Wenn man h_1 als Hashtabel benutzt ist die Wahrscheilichkeit, mit der eine Kollision auftritt auch ist, da die große Zahl gg ist und deshalb nur bis dein platz 18 kann also gegeben werden (ohne Kollision) für die H_2 müssen wir vermeiden, dass die Zahl x mit ihre Vielfach 23.x als Schlüssel ausgewählt. Ebenso kann man Kollision vermeiden.

b) I) 1) Lineare Sondieren mit H_1 als Hashfunktion

Schlüssel: 61 16 61 87 69 90 4 43 57 4 12 80 46

$h_1 * 6$ $h_1(6) = 6$ platz 6

* 16 $h_1(16) = 7$ platz 7

* 61 $h_1(61) = 7$ platz 7 besetzt $h_1(67)+1 = 8$ platz 8

* 87 $h_1(87) = 15$ platz 15

* 69 $h_1(69) = 15$ platz besetzt $h_1(69)+1 = 16$ platz 16

* 90 $h_1(90) = 9$ platz 9

* 4 $h_1(4) = 4$ platz 4

* 43 $h_1(43) = 7$ besetzt $h_1(43)+1 = 8$ besetzt $h_1(43)+2 = 9$ besetzt

$h_1(43)+3 = 10$ platz 10

* 57 $h_1(57) = 12$ platz 12

* 4 $h_1(4) =$ 4 besetzt $h_1(4)+1 = 5$ platz 5

* 12 $h_1(12) = 3$ platz 3

* 80 $h_1(80) = 8$ besetzt $h_1(80)+1 = 9$ besetzt $h_1(80)+2 = 10$

$h_1(80)+3 = 11$ platz 11

* 46 $h_1(46) = 10$ besetzt $h_1(46)+1 = 11$ besetzt

$h_1(46)+2 = 12$ besetzt

$h_1(46)+3 = 13$

platz 13

b) I) 2) Linear sondieren mit h_2 als Hashfunktion.

* 6 $h_2(6) = 6$ platz 6
* 16 $h_2(16) = 16$ platz 16
* 61 $h_2(61) = 15$ platz 15
* 87 $h_2(87) = 18$ platz 18
* 69 $h_2(69) = 0$ platz 0
* 90 $h_2(90) = 21$ platz 21
* 4 $h_2(4) = 4$ platz 4
* 43 $h_2(43) = 20$ platz 20
* 57 $h_2(57) = 11$ platz 11
* 41 $h_2(41) = 4$ besetzt $h_2(41)+1 = 5$ platz 5
* 12 $h_2(12) = 12$ platz 12
* 80 $h_2(80) = 13$ besetzt $h_2(80)+1 = 12$ besetzt $h_2(80)+2 = 13$ platz 13
* 46 $h_2(46) = 0$ besetzt $h_2(46)+1 = 1$ platz 1

b II) 1) Lin Quadratische Sondierung mit H_1 als Hash Funktion

* 6 $h_1(6) = 6$ platz 6
* 16 $h_1(16) = 7$ platz 7
* 61 $h_1(61) = 7$ besetzt $h_1(61) + 1^2 = 8$ platz 8
* 87 $h_1(87) = 15$ platz 15
* 69 $h_1(69) = 15$ ~~platz~~ besetzt $h_1(69) + 1^2 = 16$ platz 16
* 90 $h_1(90) = 9$ platz 9
* 4 $h_1(4) = 4$ platz 4
* 43 $h_1(43) = 7$ besetzt $h_1(43) + 1^2 = 8$ besetzt
  $h_1(43) + 2^2 = 11$ platz 11
* 57 $h_1(57) = 12$ platz 12
* 4 $h_1(4) = 4$ besetzt $h_1(4) + 1^2 = 5$ platz 5
* 12 $h_1(12) = 3$ platz 3
* ~~82~~ $h_1(80) = 8$ besetzt $h_1(80) + 1^2 = 9$ besetzt
  $h_1(80) + 2^2 = 12$ besetzt
  $h_1(80) + 3^2 = 17$ platz 17
* $h_1(46) = 10$ platz 10

a) II) 2) Quadratisches Sondieren mit h_2 als Hashfunktion

* 6 $h_2(6) = 6$ platz 6
* 16 $h_2(16) = 16$ platz 16
* 61 $h_2(61) = 15$ platz 15
* 87 $h_2(87) = 18$ platz 18
* 69 $h_2(69) = 0$ platz 0
* 90 $h_2(90) = 21$ platz 21
* 4 $h_2(4) = 4$ platz 4
* 43 $h_2(43) = 20$ platz 20
* 57 $h_2(57) = 11$ platz 11
* 4 $h_2(4) = 4$ besetzt $h_2(4) + 1^2 = 5$ platz 5
* 12 $h_2(12) = 12$ platz 12
* 80 $h_2(80) = 11$ besetzt $h_2(80) + 1^2 = 12$ besetzt
  $h_2(80) + 2^2 = 15$ besetzt
  $h_2(80) + 3^2 = 20$ besetzt
  $h_2(80) + 4^2 = 27 \mod 23 = 4$ besetzt
  $h_2(80) + 5^2 = 36 \mod 23 = 13$

  platz 13
* 46 $h_2(46) = 0$ besetzt $h_2(46) + 1^2 = 1$ platz 1