```
public struct Bug {
    string bez;

    public bool istDuplikat(Bug b) {
        if(bez == b.bez)
            return true;
        else
            return false;
    }
}
```

= "1"

= "2"

b2

b2.bez

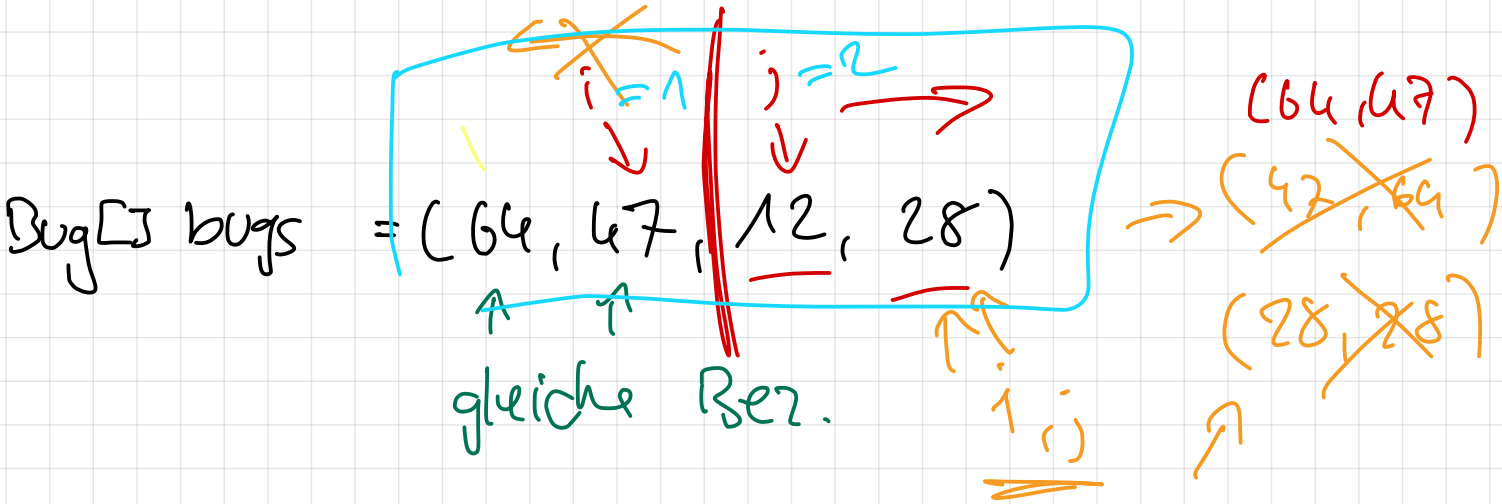
b1 = new Bug("1")

b2 Bug("2")

b1.istDuplikat(b2)

b1.Hallowelt()

Console.WriteLine("Hallo");




```
for (int i=0; ...) {
    for (int j = 0; ...) {
        if (bugs[i].isDuplicat(bugs[j])){
            ...
        }
    }
}
```

i+1

[1,2,3,4,5,6,7,8,9]

0,1,2 ... 9

Bugs.length = 10

```
for (int i=0; i < Bug.length ...)
    for (int j = i+1; j < Bugs.length ) {
        ...
    }
}
```

j = 10

10 < 10 ✗

1. Semester Informatik
IN, WIN, MIN, AMP
Programmieren I, Wintersemester 2018/19

TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Prüfer: Profs. Dres. Albrecht, Hartmann, Stappert, Wienkop
Prüfungstermin: 18.1.2019, 8.30 Uhr — Dauer: 90 Minuten
Hilfsmittel: 6 selbst erstellte Seiten — 5 Aufgaben auf 3 Aufgabenblättern
Prüfungspapier: 3 Bögen — **doppelseitig bedruckt**

Hinweise zur Bearbeitung:

- Sie dürfen mit **Bleistift** schreiben. Schreiben Sie **äußerst leserlich**!
- Sie dürfen für die Bearbeitung der Aufgaben nur die jeweils bei den Aufgaben angegebenen .NET-Bibliotheksfunktionen verwenden! Falls keine angegeben sind, müssen Sie die Funktionalität entsprechend selbst programmieren! Ausnahmen, die Sie (sofern nicht anders angegeben) verwenden dürfen, sind die Funktionen der Klassen `Console` und `Convert` sowie `Length` und `GetLength`.
- Die Verwendung von `goto` und `continue` ist nicht zulässig und führt zu Punktabzug.

Die Bearbeitung der Aufgaben 1-4 hat auf dem karierten Prüfungspapier zu erfolgen.
Die Lösung für Aufgabe 5 hingegen ist direkt auf dem Aufgabenblatt zu notieren.
Trennen Sie dafür die letzte Doppelseite ab und tragen Sie Name und Matrikelnummer ein.
Die Blätter mit Aufgaben 1-4 brauchen nicht mit abgegeben zu werden!

Viel Erfolg!

---

1 Lokale Minima [7 Punkte]

Erstellen Sie eine Funktion „`Minima`“, die ein `int`-Feld als Parameter erhält und alle lokalen Minima dieses Feldes an der `Console` ausgibt.
Ein Feldelement ist ein lokales Minimum, wenn sowohl sein linker als auch sein rechter Nachbar größer sind als es selbst.

Beispiel:
Lokale Minima in {10, 1, 20, 5, 4, 11, 3, 7, 6}:
1
4
3

2 Schriftlich Addieren [15 Punkte]

Erstellen Sie eine Funktion „`Addiere`“, die zwei ganze Zahlen beliebiger Größe addiert. Die beiden Zahlen werden als `string`-Parameter übergeben, der Rückgabewert ist ebenfalls ein `string`.
Die Funktion addiert die einzelnen Ziffern von rechts nach links – mit Berücksichtigung eines ggf. auftretenden Übertrags.
Der Einfachheit halber können Sie davon ausgehen, dass beide übergebenen Strings gleich lang sind und nur Ziffern enthalten.

Hinweis: Es muss ziffernweise addiert werden wie bei der schriftlichen Addition, Sie dürfen die Strings **nicht** einfach mit Convert.ToInt32() o.ä. in Zahlen konvertieren!

Beispiel:
"42" + "29" = "71"
"542" + "629" = "1171"

3 Klasse "Bug" zur Fehlerverwaltung [28 Punkte]

Bei professionellen Software-Projekten gibt es eine Fehlerdatenbank, in der die "Bugs" registriert werden. In dieser Aufgabe sollen Sie eine einfache Klasse `Bug` zur Verwaltung von Software-Fehlern definieren.

a) Definieren Sie einen Aufzählтyp `Status` mit den möglichen Werten `Neu, Offen, Bearbeitet, Geschlossen.`

b) Definieren Sie die Klasse `Bug` mit folgenden geschützten Datenfeldern:

- id: eine ganze Zahl, die einen Bug eindeutig identifiziert
- bez: eine Bezeichnung bzw. textuelle Beschreibung
- status: der Status des Bugs; verwenden Sie hierfür den Aufzählтyp aus a)

c) Schreiben Sie einen öffentlichen Konstruktor für die Erstellung eines neuen Bug-Objekts:

- Der Konstruktor erhält Parameter für die Bezeichnung und den Status.
  Der Status-Parameter ist optional. Falls nicht angegeben, wird er auf `Neu` gesetzt.
- Stellen Sie sicher, dass die Bezeichnung nicht leer ist.
  Anderenfalls lösen Sie eine `ArgumentException()` aus.
- Die ID soll automatisch fortlaufend vergeben werden.

d) Schreiben Sie ein öffentliches Read-Only-Property `Gefixt`, das `true` zurückliefert, wenn der Status des Bugs `Bearbeitet` oder `Geschlossen` ist, ansonsten `false`.

e) Schreiben Sie eine Methode `IstDuplikat()`, die ein Bug-Objekt `b` als Parameter erhält. Sie soll für einen Bug prüfen, ob `b` ein Duplikat zu diesem ist. Stimmt die Bezeichnung überein, wird `true` zurückliefert, ansonsten `false`.

f) Schreiben Sie unter Verwendung von `IstDuplikat()` eine Methode `DuplikateAusgeben()`.
Sie soll ein Array bzw. eine variable Anzahl von Bugs als Parameter erhalten und die ID-Paare der Bugs auf der Console ausgeben, die Duplikate zueinander sind.

Informelles Beispiel für ein Bug-Array mit 6 Bugs (ohne Status):

{ (47, "Fehler A"), (51, "Fehler XY), (56, "Fehler Z"),
(64, "Fehler A"), (82, "Fehler A"), (99, "Fehler Z") }

`DuplikateAusgeben()` soll dafür eine Ausgabe dieser Form erzeugen (Reihenfolge egal):

47 = 64
47 = 82
56 = 99
64 = 82

Achten Sie darauf, jedes Wertepaar Reihenfolge-unabhängig nur einmal auszugeben, d.h. nicht auch noch "64 = 47", wenn schon "47 = 64" ausgegeben wurde.

g) Definieren Sie in einer weiteren Klasse `Program` eine Main-Funktion, in der Sie

- drei Bugs erzeugen, davon einer mit Status `Offen`,
- die Methode `DuplikateAusgeben()` für diese drei Bugs aufrufen.

4 Dateiverarbeitung – Nächste Fähren [20 Punkte]

Die Fährverbindungen einer Reederei sind in einer Textdatei gespeichert. Rechts ist ein beispielhafter Inhalt zu sehen. In jeder Zeile steht eine Verbindung mit durch ‚|' getrennten Informationen:

- Abfahrtzeit
- Von-Nach-Information
- Preis

```
08:00|Dünkirchen nach Dover|€82,00
09:20|Calais nach Dover|€87,00
10:00|Dünkirchen nach Dover|€82,00
10:55|Calais nach Dover|€87,00
12:00|Dünkirchen nach Dover|€82,00
12:25|Calais nach Dover|€80,00
13:55|Calais nach Dover|€99,00
14:00|Dünkirchen nach Dover|88,00
15:30|Calais nach Dover|€95,00
16:00|Dünkirchen nach Dover|€90,00
17:00|Calais nach Dover|€98,00
```

Schreiben Sie eine Funktion

```
public static void NaechsteFaehren(string ort, int std, int min)
```

welche für einen angegebenen Abfahrthafen und eine Uhrzeit die nächsten bis zu drei Verbindungen auflistet.

So würde beispielsweise **`NaechsteFaehren("Calais", 10, 00)`** die folgenden Ausgaben zur Folge haben:

```
Fähre von Calais nach Dover: 10:55  €87,00
Fähre von Calais nach Dover: 12:25  €80,00
Fähre von Calais nach Dover: 13:55  €99,00
```

Sollten bis zum Dateiende weniger als drei Auflistungen möglich sein, so ist dies auch ohne weiteren Kommentar in Ordnung. Wird hingegen überhaupt keine Verbindung mehr gefunden, so ist ”Keine Fähre mehr!” auszugeben.

Hinweise:

- Die Daten befinden sich in einer Textdatei ”Fahrplan.txt”. Sie müssen keine Vorkehrungen treffen, falls die Datei nicht existiert oder nicht gefunden wird.
- Achten Sie darauf, dass unter allen Umständen die Datei geschlossen wird!
- Stunden- und Minuten-Informationen können für handlichere Vergleiche auch in Nur-Minuten umgerechnet werden!
- Sie dürfen die String-Funktionen `bool string.StartsWith(string s)` für den Ortstest und `string[] string.Split(params char[] sep)` für das Zerlegen einer Zeile gerne verwenden.

5 Code-Verständnis [20 Punkte]

a) Folgende Funktion ist gegeben:

```
static void Erzeuge (int a, int b)
{
        for (int i=1; i<=b; i++)
        {
                for (int j=a+i; j<a+b+i; j++)
                {
                        Console.Write($"{j} ");
                }
                Console.WriteLine();
        }
}
```

Welche Ausgabe liefert der Aufruf : (4 Punkte)

`Erzeuge(20,4)`

21	22	23	24
22	23	24	25
23	24	25	26
24	25	26	27

b) Folgende Funktion ist gegeben:

```
static void Funktion(params int[] arr)
{
        foreach (int i in arr)
        {
                if (i%2 == 0)
                {
                        if (i>=0 && i <= 10)
                        {
                                Console.Write ("A");
                        }
                }
                else if (i< 0 || i%3 == 0)
                {
                        Console.Write("B");
                }
                else
                {
                        Console.Write("C");
                }
}
```

Welche Ausgabe erzeugt (4 Punkte)

`Funktion(8,21,20,9)`

A B B

c) Kennzeichnen Sie folgende Aussagen mit richtig oder falsch (4 Punkte)

	Richtig	Falsch
Eine Klasse kann mehrere Konstruktoren haben	X	
Den Defaultkonstruktor (Konstruktor ohne Parameter) gibt es immer	X	
Die Änderung einer statischen Klassenvariable betrifft alle Instanzen der Klasse		X
Ein Array, als Parameter einer Funktion und mit params deklariert, darf in der Liste der Parameter nur einmal vorkommen	X	

d) Ergänzen Sie folgende Funktion, die in einem zweidimensionalen Rechteckarray die größte Zahl bestimmt und zurückliefert: (4 Punkte)

```
static int Maximum (int[,] array)
{
    int max = array[0,0];

    for (int i=0; i< array.GetLength(0); i++)
    {
        for (int j=0; j< array.GetLength(1) ;j++)
        {
            if(array[i,j] > max ){
                max = array[i,j];
            }
        }
    }
    return max ;
}
```

e) Welche Ausgabe erzeugt das folgende Programm ? (4 Punkte)

```
static void Add(out int a, ref int b)
{
    a = 3 * b;
    b -= a;
    a += b;
}
static void Main()
{
    int x, y=3;
    Add (out x, ref y);

    Console.WriteLine($"{x}  {y} ");
}
```

x	3
y	-6

Ausgabe: 3 -6