Prof. Dr. Andreas Claßen, Prof. Dr. Thomas Dey

Pflicht-Offline-Aufgabe O 06-01, INF & WI & MCD:

Zeichenkette aufteilen ab Zeichen

(geübte C++ Konstrukte: Funktionen, Referenz-parameter, `string`)

Sie brauchen für diese Offline-Aufgabe nur eine einzelne .cpp Datei zu schreiben, in der sowohl die Funktion als auch das Hauptprogramm programmiert sind. Es braucht also keine Headerdatei geschrieben zu werden und Funktionen & Hauptprogramm sollen auch nicht über mehrere Dateien verteilt werden. Dies wäre zwar schöner, würde aber den Arbeitsaufwand für das Hochladen dieser Aufgabe in den Jenkins vergrößern …

Schreiben Sie ein C++ Programm, welches eine einzeilige Zeichenkette ab dem ersten Vorkommen eines vom Benutzer eingegebenen Zeichens aufspaltet. Das Zeichen selbst soll in keinen der beiden Teile übernommen werden.

Die Berechnung soll über eine Funktion …

```
void spalte_ab_erstem(char zeichen,
                      string eingabe,
                      string& erster_teil,
                      string& zweiter_teil)
```

… realisiert werden, welche die beiden Teile mittels Referenzparametern zurückgibt.

Sollte das Zeichen nicht vorkommen, so wird die komplette Zeichenkette im ersten Ergebnis zurückgegeben und der zweite Teil ist leer.

Sollte das Zeichen mehrfach vorkommen, so wird am ersten Vorkommen des Zeichens gespalten. Alle weiteren Vorkommen des Zeichens werden wie „ganz normale" Zeichen behandelt (und sind somit Teil des zweiten Teilergebnisses).

Die einzeilige Eingabe darf auch Leerzeichen enthalten.

Hinweis: Name und Parameter/Schnittstelle der Funktion müssen exakt mit den Vorgaben oben übereinstimmen, da Jenkins ihre Funktion auch unabhängig von ihrem Hauptprogramm testet.

Testläufe (Benutzereingaben zur Verdeutlichung unterstrichen):

```
Bitte geben Sie die einzeilige Zeichenkette ein: ? 12 abcde 89
Bitte geben Sie das Zeichen ein: ? c
Der erste Teil der Zeichenkette lautet: 12 ab
Der zweite Teil der Zeichenkette lautet: de 89
Drücken Sie eine beliebige Taste . . .
```

```
Bitte geben Sie die einzeilige Zeichenkette ein: ? aaa aaa
Bitte geben Sie das Zeichen ein: ? a
Der erste Teil der Zeichenkette lautet:
Der zweite Teil der Zeichenkette lautet: aa aaa
Drücken Sie eine beliebige Taste . . .
```

```
Bitte geben Sie die einzeilige Zeichenkette ein: ? 123456
Bitte geben Sie das Zeichen ein: ? b
Der erste Teil der Zeichenkette lautet: 123456
Der zweite Teil der Zeichenkette lautet:
Drücken Sie eine beliebige Taste . . .
```

Prof. Dr. Andreas Claßen, Prof. Dr. Thomas Dey

Freiwillige Offline-Aufgabe O 06-02, INF & WI & MCD:

Addition und Subtraktion, rekursiv definiert

(geübte C++ Konstrukte: Funktionen, Rekursion)

Sie brauchen für diese Offline-Aufgabe nur eine einzelne .cpp Datei zu schreiben, in der sowohl die Funktion als auch das Hauptprogramm programmiert sind. Es braucht also keine Headerdatei geschrieben zu werden und Funktionen & Hauptprogramm sollen auch nicht über mehrere Dateien verteilt werden. Dies wäre zwar schöner, würde aber den Arbeitsaufwand für das Hochladen dieser Aufgabe in den Jenkins vergrößern …

Schreiben Sie ein C++ Programm, in welchem die Addition und Subtraktion ganzer Zahlen (Subtraktion: positiver ganzer Zahlen) rekursiv berechnet werden.

Die rekursive Definition der Addition laute dabei (ganzzahlige `i,k`):

```
addition_rekursiv(i, 0) := i
addition_rekursiv(i, k) := addition_rekursiv(i+1, k-1) für k>0
```

Die rekursive Definition der Subtraktion laute:

```
subtraktion_rekursiv(i, 0) := i
subtraktion_rekursiv(i, k) :=
                          subtraktion_rekursiv(i-1, k-1) für k>0
```

Benutzen Sie für das Programm folgendes Codegerüst:

```
#include <iostream>
using namespace std;

int addition_rekursiv(int i, int k)
{
      // … Ihr Code hier …
}

int subtraktion_rekursiv(int i, int k)
{
      // … Ihr Code hier …
}
```

```
int main()
{
      int i = 0, k = 0;
      cout << "Bitte geben Sie die erste Zahl ein: ? ";
      cin >> i;
      cout << "Bitte geben Sie die zweite Zahl ein: ? ";
      cin >> k;

      cout << i << " + " << k << " = "
           << addition_rekursiv(i, k) << endl;
      cout << i << " - " << k << " = "
           << subtraktion_rekursiv(i, k) << endl;

      system("PAUSE");
      return 0;
}
```

Hinweise:
Die Eingaben des Benutzers brauchen nicht geprüft zu werden.

Testläufe (Benutzereingaben zur Verdeutlichung unterstrichen):

```
Bitte geben Sie die erste Zahl ein: ? 11
Bitte geben Sie die zweite Zahl ein: ? 0
11 + 0 = 11
11 - 0 = 11
Drücken Sie eine beliebige Taste . . .
```

```
Bitte geben Sie die erste Zahl ein: ? 33
Bitte geben Sie die zweite Zahl ein: ? 22
33 + 22 = 55
33 - 22 = 11
Drücken Sie eine beliebige Taste . . .
```

```
Bitte geben Sie die erste Zahl ein: ? 44
Bitte geben Sie die zweite Zahl ein: ? 55
44 + 55 = 99
44 - 55 = -11
Drücken Sie eine beliebige Taste . . .
```

Prof. Dr. Andreas Claßen, Prof. Dr. Thomas Dey

Pflicht-Offline-Aufgabe 06-03, INF & WI (MCD: freiwillig):

Linien zeichnen (rekursiv)

(geübte C++ Konstrukte: Funktionen, Rekursion, Array, `for` Schleife)

Sie brauchen für diese Offline-Aufgabe nur eine einzelne .cpp Datei zu schreiben, in der sowohl die Funktion als auch das Hauptprogramm programmiert sind. Es braucht also keine Headerdatei geschrieben zu werden und Funktionen & Hauptprogramm sollen auch nicht über mehrere Dateien verteilt werden. Dies wäre zwar schöner, würde aber den Arbeitsaufwand für das Hochladen dieser Aufgabe in den Jenkins vergrößern …

Schreiben Sie ein C++ Programm, welches den folgenden, hier als „Pseudo-Code" angegebenen Algorithmus mittels einer Funktion …

```
void linie(int x1, int y1, int x2, int y2,
           char canvas[][canvas_size])
```

… realisiert.

Pseudo-Code:

```
void linie(int x1, int y1, int x2, int y2,
           char canvas[][canvas_size])
{
    if ( (x1, y1) und (x2, y2) sind benachbart ) {
        Zeichne die Punkte (x1, y1) und (x2, y2)
    }
    else {
        // Berechne die ganzzahligen Koordinaten des
        // Punktes in der Mitte zwischen den beiden
        // Ausgangspunkten:
        int x_mitte = (x1 + x2)/2;
        int y_mitte = (y1 + y2)/2;

        // Rekursive Aufrufe:
        1. Linie vom ersten Punkt bis zur Mitte
        2. Linie von der Mitte bis zum zweiten Punkt
    }
}
```

Wie man berechnet, dass zwei Punkte `(x1, y1)` und `(x2, y2)` benachbart sind (d.h. direkt nebeneinander, direkt übereinander oder direkt diagonal zueinander liegen), müssen Sie selbst herausfinden und als C++ Code codieren.

Ein Punkt wird an der Stelle `(x, y)` gezeichnet, indem das Zeichen `filled_pixel` in das Array `canvas[x][y]` eingetragen wird.

Beachten Sie, dass wegen der Indexzählung im Array ab Null auch die Koordinatenzählung ab Null beginnt, d.h. die linke obere Ecke des Diagramms hat die Koordinate `(0, 0)`.

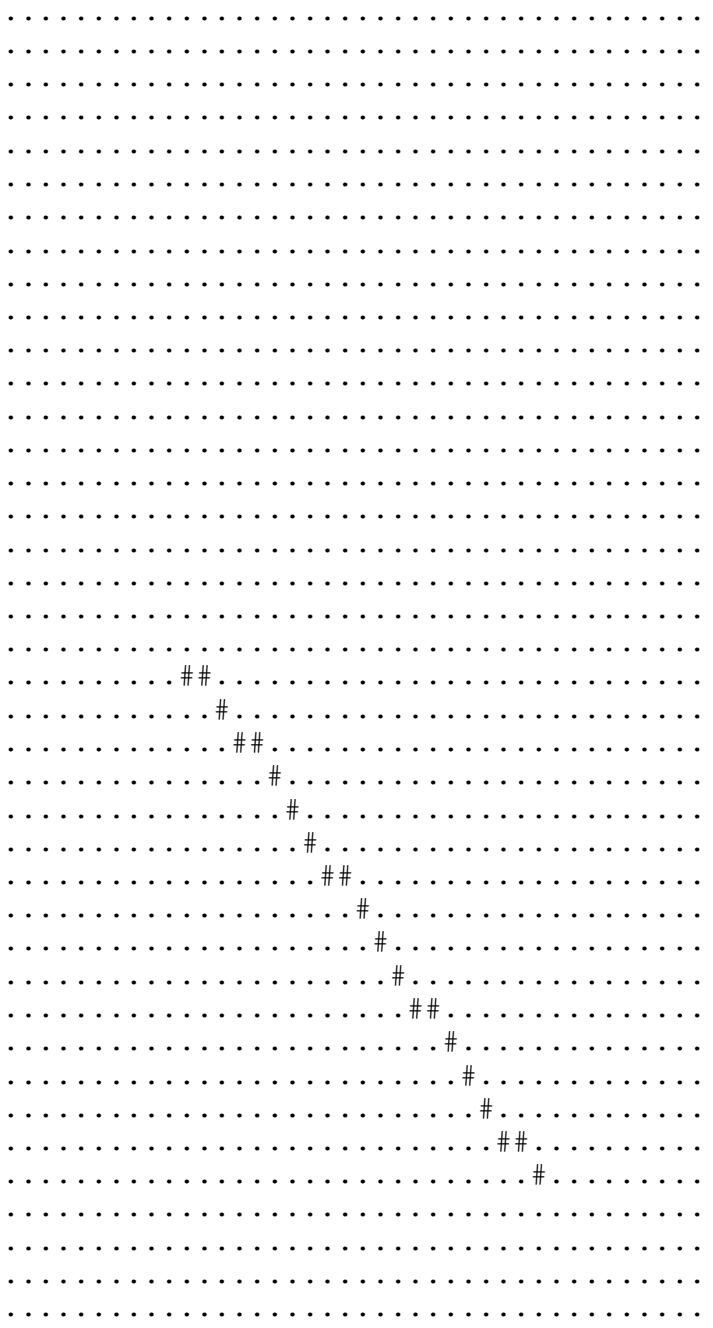
Benutzen Sie für das gesamte Programm folgendes vorgegebenes Codegerüst, welches Sie nicht verändern dürfen:

```
#include <iostream>
using namespace std;

const char empty_pixel = '.';
const char filled_pixel = '#';

const int canvas_size = 40;

void init_canvas(char canvas[][canvas_size])
{
	for (int x = 0; x < canvas_size; x++)
		for (int y = 0; y < canvas_size; y++)
			canvas[x][y] = empty_pixel;
}

void print_canvas(char canvas[][canvas_size])
{
	for (int y = 0; y < canvas_size; y++) {
		for (int x = 0; x < canvas_size; x++) {
			cout << canvas[x][y];
		}
		cout << endl;
	}
	cout << endl;
}

void linie(int x1, int y1, int x2, int y2, char canvas[][canvas_size] )
{
	// … Ihr Code hier …
}
```

```
int main()
{
	char canvas[canvas_size][canvas_size];
	init_canvas(canvas);

	int x1 = 0, y1 = 0, x2 = 0, y2 = 0;
	do {
		cout << "Bitte geben Sie den ersten Punkt ein: ? ";
		cin >> x1 >> y1;
	} while (x1 < 0 || x1 >= canvas_size || y1 < 0 || y1 >= canvas_size);
	do {
		cout << "Bitte geben Sie den zweiten Punkt ein: ? ";
		cin >> x2 >> y2;
	} while (x2 < 0 || x2 >= canvas_size || y2 < 0 || y2 >= canvas_size);

	linie(x1, y1, x2, y2, canvas);

	print_canvas(canvas);

	system("PAUSE");
	return 0;
}
```

Testläufe (Benutzereingaben zur Verdeutlichung unterstrichen):

```
Bitte geben Sie den ersten Punkt ein: ? -10 20
Bitte geben Sie den ersten Punkt ein: ? 10 20
Bitte geben Sie den zweiten Punkt ein: ? 30 90
Bitte geben Sie den zweiten Punkt ein: ? 30 35
```



```
Drücken Sie eine beliebige Taste . . .
```

```
Bitte geben Sie den ersten Punkt ein: ? 7 9
Bitte geben Sie den zweiten Punkt ein: ? 3 5
........................................
........................................
........................................
........................................
........................................
...#....................................
....#...................................
.....#..................................
......#.................................
.......#................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................

Drücken Sie eine beliebige Taste . . .
```

```
Bitte geben Sie den ersten Punkt ein: ? 2 5
Bitte geben Sie den zweiten Punkt ein: ? 35 20
........................................
........................................
........................................
........................................
........................................
..####..................................
......##................................
........##..............................
..........##............................
............##..........................
..............##........................
................##......................
..................##....................
....................##..................
......................##................
........................##..............
..........................##............
............................##..........
..............................##........
................................###.....
..................................#.....
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................

Drücken Sie eine beliebige Taste . . .
```