Aufgabe 1. Suchen Sie sich eines Ihrer bisher geschriebenen Programme aus und werfen Sie, wie in der Vorlesung gezeigt, den Debugger `gdb` darauf an. Durchlaufen Sie insbesondere Ihr Programm schrittweise mit dem Debugger und lassen Sie sich den Wert einzelner Variablen ausgeben.

Aufgabe 2. Unten angehängt (aber auch auf der Veranstaltungshomepage zum Herunterladen) finden Sie eine Klasse, die eine Matrix speichert. Erweitern Sie diese Klasse, indem Sie mit `try{ ... }` und `catch` an allen Stellen, an denen möglicherweise ein Zugriff auf einen falschen Speicherbereich stattfindet, passende Überprüfungen durchführen. Fügen Sie auch einen Konstruktor ein, der eine quadratische Einheitsmatrix erzeugt. Schreiben Sie außerdem einen +-Operator und eine Methode, welche die transponierte Matrix zurückgibt, die also zu einer $m \times n$-Matrix eine $n \times m$-Matrix berechnet, in welcher für alle passenden Indizes i und j der Eintrag an der Stelle (i, j) gerade der Eintrag an der Stelle (j, i) der ursprünglichen Matrix ist.

Aufgabe 3. Schreiben Sie ein Programm, das im selben Format wie in Aufgabe 2 des vorigen Zettels eine Menge von achsenparallelen Rechtecken einliest und dann überprüft, ob es einen Punkt gibt, der von mindestens vier dieser Rechteckecke überdeckt wird. Vergeichen Sie die Laufzeit Ihres Programms mit der Ihrer Lösung von Aufgabe Aufgabe 2 des vorigen Zettels. Erzeugen Sie mittels eines Programms auch selbst Eingabedateien (indem Sie die Ausgabe eines Programms mit > in eine Datei schicken).

Aufgabe 4. Schreiben Sie ein Programm, dem per Kommandozeilenargument zwei Dateinamen übergeben werden. Die erste Datei soll wieder eine Menge von Rechtecken kodieren (Format: siehe Aufgabe 2, Zettel 8). Die zweite Datei soll eine Menge von Punkten in der Ebene kodieren, wobei jeder Punkt duch eine Zeile gegeben sei, in der die x-Koodinate und die y-Koordinate des Punktes gespeichert sind. Anschließend sollen Sie überprüfen, ob es für jeden Punkt aus der zweiten Datei ein Rechteck aus der ersten Datei gibt, das ihn enthält.
Lassen Sie auch für dieses Problem Instanzen durch ein C++-Programm erzeugen.

```cpp
#include <iostream>
#include <vector>

typedef std::vector<double> zeile;

class Matrix
{
public:
   Matrix(unsigned m, unsigned n):
      _num_rows(m),
      _num_columns(n),
      _entries(m)
   {
      for(unsigned i = 0; i < _num_rows; i++)
      {
         for(unsigned j = 0; j < _num_columns; j++)
         {
            (_entries[i]).push_back(0.0);
         }
      }
   }
   double get_entry(unsigned i, unsigned j) const {
      return _entries[i][j];
   }
   unsigned get_num_rows() const {
      return _num_rows;
   }
   unsigned get_num_columns() const {
      return _num_columns;
   }
   double set_entry(unsigned i, unsigned j, double value) {
      return _entries[i][j] = value;
   }
   void ausgeben() {
      std::cout << _num_rows << " Zeilen und " << _num_columns << " Spalten:" << std::endl;
```

```
36       for(unsigned i = 0; i < _num_rows; i++)
37       {
38          for(unsigned j = 0; j < _num_columns; j++)
39          {
40             std::cout << _entries[i][j] << " ";
41          }
42          std::cout << std::endl;
43       }
44    }
45    Matrix operator* (const Matrix & other) const {
46       Matrix result(_num_rows, other._num_columns);
47       for(unsigned i = 0; i < _num_rows; i++) {
48          for(unsigned j = 0; j < other._num_columns; j++)
49          {
50             double entry = 0;
51             for(unsigned k = 0; k < _num_rows; k++) {
52                entry += _entries[i][k] * other._entries[k][j];
53             }
54             result.set_entry(i, j, entry);
55          }
56       }
57       return result;
58    }
59 private:
60    unsigned  _num_rows;
61    unsigned  _num_columns;
62    std::vector<zeile> _entries;
63 };
64
65
66 int main()
67 {
68    Matrix meine_matrix(3,4);
69    Matrix zweite_matrix(4,7);
70    meine_matrix.set_entry(2, 2, 5.0);
71    meine_matrix.ausgeben();
72
73    zweite_matrix.set_entry(2, 4, 3.0);
74    zweite_matrix.ausgeben();
75
76    Matrix dritte_matrix = meine_matrix * zweite_matrix;
77    dritte_matrix.ausgeben();
78    return 0;
79 }
```

Listing 1: Eine Klasse, die Matrizen verwaltet.