UNIVERSITÄT DES SAARLANDES
Fachbereich 14 – Informatik
Lehrstuhl Prof. Smolka
Tobias Müller

C–Blockkurs Übungsblatt 10

1. Aufgabe: blindes TicTacToe

TicTacToe ist ein Spiel, das auf einem drei mal drei Felder großen Spielfeld von zwei Spielern gespielt wird. Es werden abwechselnd Kreuze und Kreise eingezeichnet, bis einer der Spieler eine Zeile, Spalte oder Diagonale mit drei Zeichen gefüllt hat (Wer das Spiel nicht kennt, der sollte sich mal den Film Wargames ansehen).

Beim blinden TicTacToe ist das Spielfeld nicht sichtbar und die Spieler wählen einfach ein Feld aus, auf das sie setzen wollen. Außerdem gibt es noch den Spielleiter, der das Spielfeld verwaltet und die Züge für die Spieler aufzeichnet. Wenn ein Spieler auf ein freies Feld setzen will, so zeichnet der Spielleiter das entsprechende Zeichen ein und der nächste Spieler ist am Zug. Versucht der Spieler auf ein schon besetztes Feld zu setzen, dann teilt ihm der Spielleiter mit, daß dieser Zug ungültig war und der Spieler darf es direkt nocheinmal versuchen.

Ziel dieser Aufgabe ist es, eine Funktion `int btttSS99_x(int valid, int reset)` zu schreiben, die das Verhalten eines blinden TicTacToe Spielers simuliert. `btttSS99` steht hierbei für blindes TicTacToe im Sommersemester 99 und das x muß durch die Zahl Ihres Login-Namens ersetzt werden, z.B.: ckurs99 schreibt die Funktion btttSS99_99.

Der Spielleiter ist in unserem Fall ein Programm, das Ihre Funktionen aufruft und das Spielfeld verwaltet. Das Spielfeld soll hier durch Zahlen von 0 bis 8 repräsentiert werden. Die 0 bezeichnet das linke obere, die 2 das rechte obere und die 8 das rechte untere Feld. Jedesmal, wenn Ihre Funktion aufgerufen wird, erwartet der Spielleiter von Ihrer Funktion einen Rückgabewert, der einem Spielfeld entspricht, auf das Sie setzen wollen. War der Zug gültig, dann wird Ihre Funktion beim nächsten mal mit `valid=1` aufgerufen, ansonsten mit `valid=0`, was bedeutet, daß das letzte Feld, auf das Sie setzen wollten, schon belegt war, und daß der letzte Zug ungültig ist.

Da wir mehrere Spiele hintereinander spielen, kann es nötig sein, der `bttt` Funktion mitzuteilen, wann ein neues Spiel beginnt. Dies wird mit dem `reset` Argument realisiert. Vor jedem neuen Spiel wird `reset` ≥ 1 gesetzt, sonst 0. Nach einem Resetsignal muß Ihre Funktion eine -1 zurück geben, keine Feldposition, das Spiel hat dann noch nicht begonnen. Das `reset` Argument hat noch eine weitere Funktion, es teilt Ihnen mit, ob Sie das letzte Spiel gewonnen, verloren oder unentschieden gespielt habt:

$$\texttt{reset} = \begin{cases} 1 & : \text{unentschieden} \\ 2 & : \text{gewonnen} \\ 3 & : \text{verloren} \end{cases}$$

Die Funktionen werden von uns in das Spielleiter-Hauptprogramm eingebunden und werden viele Millionen Male gegeneinander spielen. Dabei werden Spiele gegen einen bestimmten Gegner immer am Stück gespielt. Die Funktion muß solange ziehen bis entweder ein Sieger feststeht oder keine Felder mehr frei sind. Wenn sie nach neun Versuchen keinen gültigen Zug abgegeben hat, wird dies als Kapitulation angesehen

(Das Spielfeld hat nur 9 Felder, also sollte man nach 9 Zügen ein freies Feld gefunden haben!).
Schreiben Sie Ihre Funktion in eine separate `btttSS99_x.c` Datei und den Funktionsprototyp in eine `btttSS99_x.h` Datei, so daß sie von uns einfach eingebunden werden können. *Wichtig: Die Funktion sowie das Headerfile und das Sourcefile müssen (bis auf die Endung) denselben Namen haben!*

Beispiel: **btttSS99_42** – ein randomisierter bttt – Spieler:

```
/* btttSS99_42.c */
#include <stdio.h>
#include <stdlib.h>          /* include fuer rand() */
#include "btttSS99_42.h"   /* das haederfile zu dieser .c Datei */

int btttSS99_42(int valid, int reset)
{
  int feld;                     /* Nummer des zu belegenden Feldes */
  static int vorheriges_feld=9; /* Nummer des im vorherigen Zug
                                   belegten Feldes */
                                /* Am Anfang unbekannt, darum =9 */
  if (reset)
  {
    vorheriges_feld=9;
    return -1;           /* WICHTIG! Direkt zurueckspringen */
  }

  feld = rand()%9;       /* neue Nummer erzeugen */
  return feld;           /* die Nummer des Feldes wird zurueckgegeben */
}
```