Übung 7

Algorithmen und Programmierung

Jonas Henschel, Jens Pönisch, Dominik Gorgosch, Bastian Felix Bachmann, Arvid Horn, Billy Naumann

Aufgabe 1 (Übungsaufgabe — Entlein)

Schreiben Sie ein Programm, das die Bewegungen einer Ente auf einem Spielfeld simuliert. Das Spielfeld wird durch ein zweidimensionales Array repräsentiert, wobei jedes Feld Nahrung (`FOOD`) oder keine Nahrung (`EMPTY`) enthält. Das Programm wird mit 3 Argumenten aufgerufen, das erste stellt die X-Dimensionen des Spielfelds, das zweite die Y-Dimensionen des Spielfelds und das dritte einen String dar, der angibt, ob das zweidimensionale Array am jeweiligen Index Nahrung oder keine Nahrung enthält. Zum Beispiel:

```
./entlein 3 3 FFFEEEEEE
```

Die Ente startet auf einem bestimmten Feld, dessen Koordinaten über `stdin` eingelesen werden sollen. Außerdem soll der Hunger der Ente über `stdin` eingelesen werden. Zur Steuerung der Ente gibt es folgende valide Befehle: U - Up, D - Down, L - Left, R - Right. Eine Eingabe könnte so aussehen:

```
0 0 2 DLLRUR
```

Das Entlein beginnt in diesem Beispiel seine Reise bei den Koordinaten $(0, 0)$ (links oben – ja, das Koordinatensystem ist etwas verdreht) und hat einen Hungerwert von 2. Der Hungerwert besagt, dass das Entlein Zwei Nahrungsfelder besuchen muss, um satt zu werden. Das Entlein kann bereits am Startfeld fressen, wenn dieses kein leeres Feld ist. Es sind folgende enums und structs gegeben.

```
typedef enum {FOOD, EMPTY} field_t;
typedef enum {SUCCESS, WRONG_NUMBER_OF_ARGUMENTS,
   WRONG_INPUT_SIZE, INVALID_MOVE} error_t;

typedef struct
{
    size_t x;
    size_t y;
} coord_t;

typedef struct
{
    coord_t position;
    int hunger;
}duck_t;
```

a) Schreiben Sie eine Funktion, die sicherstellt, dass die Dimensionen des Spielfelds korrekt sind, basierend auf der Größe des Eingabe-Strings, der als Kommandozeilenparameter übergeben wurde. Schlägt die Überprüfung fehl, soll das Programm mit dem Fehler `WRONG_INPUT_SIZE` beendet werden. Folgende Funktionssignatur ist gegeben:

```
1 bool check_input_size(size_t x_dim, size_t y_dim,
    char const* input)
```

b) Schreiben Sie eine Funktion, die das Spielfeld mit Nahrung und leeren Feldern gemäß dem Eingabe-String initialisiert. Folgende Funktionssignatur ist gegeben:

```
1 void fill_game_board(size_t x_dim, size_t y_dim,
    field_t game_board [x_dim][y_dim], char const *
    input)
```

c) Schreiben Sie eine Funktion, die das aktuelle Spielfeld auf der Konsole ausgibt. Folgende Funktionssignatur ist gegeben:

```
1 void print_game_board(size_t x_dim, size_t y_dim,
    field_t game_board[x_dim][y_dim])
```

d) Schreiben Sie eine Funktion, die die Bewegung der Ente gemäß der eingegebenen Richtungsanweisung auf Verletzung der Feldgrenzen überprüft. Bei Verletzung der Spielfeldgrenzen wird `INVALID_MOVE` zurückgegeben und die Ente bleibt auf ihrem Feld stehen. Ansonsten gibt die Funktion `SUCCESS` zurück und aktualisiert die Position der Ente. Folgende Funktionssignatur ist gegeben:

```
1 error_t move(char instr, duck_t *duck, size_t x_dim
    , size_t y_dim)
```

e) Schreiben Sie eine Funktion, die überprüft, ob der eingegebene Bewegungsbefehl valide ist. Ein Bewegungsbefehl ist valide, wenn es einer der Buchstaben U, D, L oder R ist. Ihre `main`-Funktion soll so lange Eingaben von `stdin` einlesen, bis ein nicht valider Befehl eingegeben wird. Folgende Funktionssignatur ist gegeben.

```
1 bool is_valid(char instr)
```

f) Schreiben Sie eine Funktion, die überprüft, ob die Ente auf einem Nahrungsfeld steht, und entsprechend den Hunger der Ente verringert und das Feld leert. Folgende Funktionssignatur ist gegeben:

```
1 void eat(duck_t *duck, size_t x_dim, size_t y_dim,
    field_t game_board[x_dim][y_dim])
```

g) Schreiben Sie eine Funktion, die überprüft, ob die Ente noch hungrig ist. Folgende Funktionssignatur ist gegeben:

```
1 bool still_hungry(duck_t duck)
```

Aufgabe 2 (Übungsaufgabe — World of Pawcraft)

World of Pawcraft ist ein Online-Rollenspiel, bei dem Ihr Charakter alltägliche Aufgaben erledigt (Wäsche bügeln, kochen, für AuP lernen) und dabei jedoch von verschiedensten Katzen (Ägyptische Mau, American Wirehair, Belinese, Bengalkatze) angegriffen werden kann. Katzen besitzen dabei unterschiedliche Attacken (fauchen, kratzen, anspringen).

Wenn Ihr Charakter angegriffen wird, unterbricht er seine aktuelle Aufgabe und geht in den Verteidigungsmodus über, in dem er sich gegen die Katze wehrt und sie letztendlich verscheucht.

Wenn Ihr Charakter zwei Aufgaben erledigt oder nachdem er eine Katze vertrieben hat, wird er müde und muss schlafen, bis sein Wecker klingelt. Danach widmet er sich wieder seinen Aufgaben. Eine Katze kann auch angreifen, während Ihr Charakter schläft: dann wacht er auf, wehrt sie ab und geht anschließend wieder schlafen.

a) Wie könnte man den Tagesablauf eines Nutzers modellieren?

b) Implementieren Sie Ihr Modell mit entsprechenden Kommandozeilenausgaben und Nutzereingaben in C.