1. házi feladat: Elixir

Követelmények

A honlapon kiírt követelmények alapján a *Tent and Trees* feladványt kell megvalósítani. Ehhez három publikus függvény létrehozása az elvárás:

- to_internal/1
- to_external/3
- check_sol/2

to_internal/1

Bemeneti paramétere egy fájlnév, amiből az aktuális feladványt beolvassa. A később szükséges információkat három listában adja vissza:

- a sátrak soronkénti darabszámát tartalmazó lista
- a sátrak oszloponkénti darabszámát tartalmazó lista
- a fákat tartalmazó parcellák koordinátáit tartalmazó lista

A beolvasandó fájl formátuma:

```
   1  0  2  0  2
1  -  *  -  -  -
1  -  -  -  -  -
0  -  -  *  -  *
3  -  -  -  -  -
0  *  -  -  -  *
```

A várt kimeneti formátum a következő:

```
{ [1, 1, 0], [1, 0, 2], [{1, 2}, {3, 3}] }
```

to_external/3

A függvény bemeneti paraméterei a to_internal/1 által feldolgozott információk hármasa, a sátrak irányát meghatározó lista és egy fájlnév. Ezek alapján a megadott fájlba létre kell hoznia a megfelelő táblázatot, majd az **:ok** atommal visszatérni.

A kapott paraméterek formátuma:

```
{ [1, 1, 0], [1, 0, 2], [{1, 2}, {3, 3}] }, [:N, :W, :S], "file.txt"
```

A várt kimeneti fájl formátuma a következő:

```
    1  0  2  0  2
  1 -  *  E  -  -
  1 -  -  -  -  N
  0 -  -  *  -  *
  3 N  -  S  -  N
  0 *  -  -  -  *
```

check_sol/2

A to_external/3 első két bemeneti paraméterével azonos adatokat kap meg. Ez egy potenciális megoldás, amely nem feltétlenül helyes. A függvénynek ezt ellenőriznie kell, és a felmerülő hibákat egy négyesben jelezni. Hibának számítanak a következők:

- különbözik egymástól a sátrak és a fák száma (nem minden fához tartozik sátor és fordítva)
- a sátrak száma nem megfelelő az adott sorokban
- a sátrak száma nem megfelelő az adott oszlopokban
- a felsorolt koordinátájú sátrak másikkal érintkeznek (oldalukon vagy sarkukon)

A bemeneti paraméterek formátuma:

```
{ [1, 1, 0], [1, 0, 2], [{1, 2}, {3, 3}] }, [:N, :W, :S]
```

A várt kimeneti formátum a következő:

```
{
  %{ err_diff: [{5, 6}] },
  %{ err_rows: [4] },
  %{ err_cols: [] },
  %{ err_touch: [{4, 3}, {5, 4}] }
}
```

Pontosított feladatspecifikáció

A Tent and Trees játékról

Egy pálya a következőképpen néz ki:

	1	**0**	**-1**	**0**	**2**
1	-	*	E	-	-
1	-	-	-	-	N
-1	-	-	*	-	*
3	N	-	S	-	-
0	*	-	-	W	*

- A felső sor adja meg, hogy az adott oszlopnak hány sátrat kell tartalmaznia.
- A bal oszlop adja meg, hogy az adott sornak hány sátrat kell tartamaznia.
- A -1 határozatlan számú sátrat jelöl az adott sorban/oszlopban.
- Jelölések:
    - üres mező **-**
    - fa *****
    - sátor **N, E, W, S**
- Minden fához pontosan egy sátornak kell tartoznia, tehát közvetlen oldalszomszédja kell, hogy legyen.
- Hogy mely fa-sátor párok tartoznak össze a táblán, azt az égtájak (N, E, W, S) határozzák meg (pl. N sátor a fájától északra található).

Tervezés (eljárások, függvények specifikálása)

A to_internal/1 megvalósításához a következő privát függvényeket hozom létre:

to_rows/1

Egy kapott listát a sorvége karakter mentén több listára bontja.

spaces/1

A kapott beágyazott listákból eltávolítja a szóköz-jellegű karaktereket.

trim/1

Eltávolítja az üres elemeket a kapott listából.

parse/1

A kapott karakterlistát integerek listájává alakítja.

row_nums/1

Visszaadja a sorokhoz tartozó elvárt sátrak darabszámának listáját.

field_coords/1

Meghatározza a fát jelölő karakterek koordinátáját a mátrixban, párok listáját ad vissza.

remove_nil/1

A kapott listából eltávolítja a nil elemeket.

A to_external/3 megvalósításához a következő privát függvényeket hozom létre:

edit_cols/1

Az oszlopokban megadott sátorszámot az elvárt módon formázza.

calc_tent/3

A fák és égtájak listája alapján visszatér a sátrak koordinátáinak listájával.

place_tent/3

A kapott fasor, faoszlop, égtáj paraméterek alapján kiszámolja a sátor helyét és *{sor, oszlop, égtáj}* hármas formájában adja vissza.

draw_fields/4

Kirajzolja a játéktáblát a kapott adatok alapján. Paraméterei a sorokban és oszlopokban elvárt sátrak darabszámai, a fák koordinátáinak listája és a sátrak koordinátáinak listája.

A check_sol/2 megvalósításához a következő privát függvényeket hozom létre:

e_diff/2

Megvizsgálja, hogy a sátrak és fák száma azonos-e.

e_cols/2

Visszaadja azon oszlopok sorszámát, amelyekben nem megfelelő számú sátor található.

e_rows/2

Visszaadja azon sorok sorszámát, amelyekben nem megfelelő számú sátor található.

e_touch/2

Visszaadja a más sátrakkal érintkező sátrak koordinátáinak listáját.

Algoritmusok

to_internal algoritmusa

A kapott fájlt beolvassuk, majd a sorvége karakterek mentén listákra bontjuk. Ezekből kiszűrjük a szóköz-szerű karaktereket, az üres szavakat és sorokat. Az első sort leválasztjuk és formázzuk, ez tartalmazza az oszlopok sátorszámát. A lista farkát továbbítjuk a row_nums/1 függvénynek, amely a sorok első szavaival tér vissza, majd ezen ereménylista szavait integerekké konvertáljuk a parse/1 függvénnyel. Már csak a fák koordinátáit kell megadnunk, ezeket a field_coords/1 függvény fogja meghatározni. Az ereménylista tartalmaz nil-t is, ezeket még el kell távolítanunk. Végül visszatérünk a **{ [sátor/sor], [sátor/oszlop], [fák koordinátái]}** hármassal.

```
 def to_internal(file) do
    file = File.read!(file)
    file = Enum.filter(trim(file), fn z -> z != [] end)
    [h | t] = file
    col = parse(h)
    row = row_nums(t) |> parse()
    field = field_coords(t) |> remove_nil() |> List.flatten()
    {row, col, field}
  end

  defp row_nums(rows) do
    for row <- rows do
      [h | _] = row
      h
    end
  end
```

field_coords algoritmusa

A beolvasott fájl sorait kapja meg listák listájaként, szavakra bontva, ebből meghatározza a fák koordinátáit. Először elhagyjuk a sor sátorszámát meghatározó első szót, majd a lista farkának minden elemét megvizsgáljuk. Ha valamelyik ***** , akkor a koordinátáit visszaadjuk egy párban.

```
defp field_coords(rows) do
  for row <- Enum.with_index(rows) do
    {[_ | t], i} = row
    for x <- Enum.with_index(t) do
      {r, j} = x
      if r == "*", do: {i + 1, j + 1}
    end
  end
end
```

calc_tent és place_tent algoritmusa

A calc_tent a fák és az égtájak listáját kapja meg. A fák listáján végighaladva minden elemhez kiválasztja a hozzá tartozó égtájat. Ezeket továbbítja a place_tent függvénynek, ami az égtájnak megfelelően beállítja a sátor koordinátáit és egy hármasban {sor, oszlop, égtáj} adja vissza. Ebből a calc_tent kiszűri az esetleges pályán kívül eső sátrakat. Mivel a fa és az égtáj lista azonos hossza nem garantált, a default kimenet {sor, oszlop, X} eredményt ad. (Ezeket a kirajzoláskor végül figyelmen kívül hagyjuk, l. draw_fields.)

```
defp calc_tent(fs, ds, max_coord) do
    {max_x, max_y} = max_coord

    tents = for x <- Enum.with_index(fs) do
      {{row, col}, i} = x
      place_tent(row, col, Enum.at(ds, i))
    end

    Enum.filter(tents, fn {x, y, _} ->
      x > 0 and y > 0 and x <= max_x and y <= max_y
    end)
  end

defp place_tent(row, col, dir) do
  case dir do
    :N -> {row - 1, col, :N}
    :E -> {row, col + 1, :E}
    :S -> {row + 1, col, :S}
    :W -> {row, col - 1, :W}
    _ -> {row, col, :X}
  end
end
```

draw_fields algoritmusa

Bemeneti paraméterként megkapja a sorok (*rs*) és oszlopok (*cs*) sátorszámának listáját, a fák (*fs*) és sátrak (*tents*) koordinátáit. A sorlista és azon belül az oszloplista minden elemére megkeresi az illeszkedő első elemet a falistából (ahol *x* a sorlista aktuális indexe és *y* az oszloplista aktuális indexe). Ha talál, akkor kirajzolja a fát jelölő ***** -t, ha ilyen nincsen, ugyanezt elvégzi a sátorlistára is. Ha ott talál, akkor kirajzolja a sátrat megadó hármas utolsó tagját (égtáj). Ha egyik listában sem talál illeszkedő elemet, akkor üres mezőt rajzol (**-**). _Megj. Ha az égtájlista rövidebb volt és a sátorlista **{x, y, :X}** elemeket tartalmaz, ez nem okoz hibát, mivel ilyenkor az adott koordinát hamarabb megtaláljuk a fa listában. Ha az égtájlista hosszabb volt, ez szintén nem okoz problémát, mivel ekkor **{nil, nil, :égtáj}** hármasokat kapunk, amelyek koordináták híján nem jönnek szóba a kirajzoláskor.

```
for row <- Enum.with_index(rs) do
  {r, i} = row

  col =
    for col <- Enum.with_index(cs) do
      {_, j} = col
      if(Enum.find(ts, fn {x, y} -> x == i + 1 and y == j + 1 end) != nil) do
        " * "
      else
        f = Enum.find(tents, fn {x, y, _} -> x == i + 1 and y == j + 1 end)
        if(f != nil) do
          {_, _, z} = f
          " #{z} "
        else " - "
        end
      end
    end
    if(r < 0) do
      ["#{r} " | col ++ ["\n"]]
    else
      [" #{r} " | col ++ ["\n"]]
    end
end
```

e_rows algoritmusa

Az e_rows függvény visszaadja azon sorok sorszámát, amelyekben nem megfelelő számú sátor található. A számlista minden elemére leszűri a sátorlista azon elemeit, amelyek *y* koordinátája megegyezik az adott sor sorszámával. Ha ennek a szűrt listának az elemeinek összege nem egyezik meg az sor sátorszámával, akkor hozzáadjuk a hibás oszlopok listájához. Az e_cols működése is hasonló.

```
  for r <- Enum.with_index(rs) do
    {r, i} = r
    if(r > -1) do
      f = Enum.filter(tents, fn {x, _, _} -> x == i + 1 end)
      if(Enum.count(f) != r) do
        i + 1
      end
```

e_touch algoritmusa

A sátrak koordinátáinak listáját kapjuk meg. Ebből először kiszűrjük a táblán kívül eső sátrakat. A kapott eredménylista minden elemére megvizsgáljuk, hogy a sátor szomszédos mezőinek a fele tartalmaz-e másik sátrat. Ha egy sátor például az ábrán látott W pozícióban van, akkor az ütközést a W sátor vizsgálatakor felfedezzük. (Így nem kerülnek duplán az ütközési listába és feleannyi elemet elég megvizsgálnunk.)

	0	**0**	**0**	**0**	**0**
0	-	W	-	-	-
0	-	-	**N**	×	-
0	-	×	×	×	-

Jelölés:

- Aktuálisan vizsgált sátor **N**
- Megvizsgált mezők **×**

```
defp e_touch(tents, max_coord) do
  {max_x, max_y} = max_coord

  for t <- tents do
    {a, b, _} = t

    pref =
      Enum.filter(tents, fn {x, y, _} ->
        x > 0 and y > 0 and x <= max_x and y <= max_y
      end)

    f =
      Enum.filter(pref, fn {x, y, _} ->
        (x == a and y == b + 1) or (x == a - 1 and y == b + 1) or
        (y == b and x == a + 1) or (x == a + 1 and y == b + 1)
      end)

    if(Enum.count(f) != 0) do
      c =
        for coord <- f do
          {x, y, _} = coord
          {x, y}
        end

      [{a, b} | c]
    end
  end
end
```

Tesztelés

A tesztelést a kiadott példák alapján végeztem el.

Kipróbálási tapasztalatok

A program az eddigi tapasztalatok alapján az elvártnak megfelelően működik. A korábbi beadások során főként formázással kapcsolatos és a negatív sátorszámok eltérő értelmezése miatt léptek fel hibák. A beadórendszer módosítása után a pályán kívül eső sátrak szűrésével kapcsolatban volt szükség kiegészítésre.