JavaScript assignment (home project)

Submit Assignment

Due Nov 13 by 11:59pm **Points** 20 **Submitting** a file upload **File Types** zip
Available until Nov 27 at 11:59pm

Web programming JavaScript home assignment — Light bulb placement

Once upon a time, King Unnamed of Nowhereland was both smiling and crying at the same time. He was smiling because his enormous brand-new palace with many spacious rooms and corridors has just been finished. He is also crying as these rooms need to be illuminated and kept warm, but the ongoing increase of utility costs affect him as well. So now it's time to think about the placement of the palace's light bulbs. We need to place them so that everything is properly lit, but we cannot install any unnecessary bulbs.

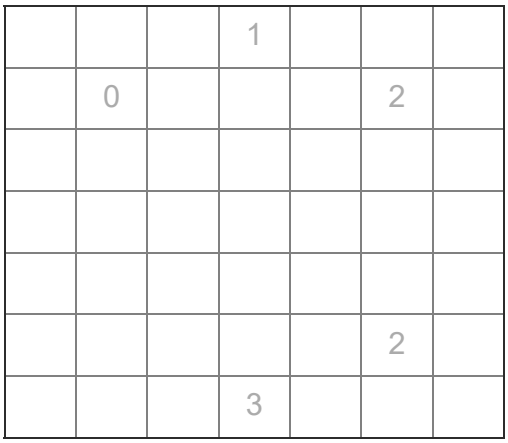
Game description

- The king's palace has rooms with square shaped floors that consist of black and white tiles only.
- Light bulbs can only be placed above white tiles.

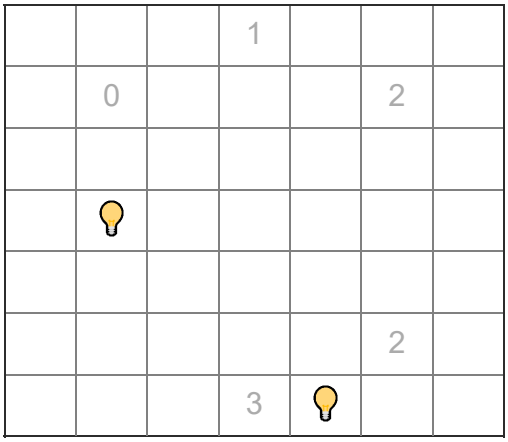
- The light from the light bulbs does not spread diagonally, only straight along the given row and column.
- The black tiles have objects placed on them, which block the propagation of light.
- Black cells can optionally contain an integer from 0 to 4. This indicates how many adjacent (bottom, top, right, left) cells contain light bulbs. If there is such a number, the puzzle must be solved accordingly!
- Two light bulbs can NEVER illuminate each other!
- The goal of the game is to place the light bulbs so that all the white tiles are illuminated.
- The game is played by one player until he solves the puzzle, so there is no need to manage multiple players at the same time or divide into rounds.

Gameplay example

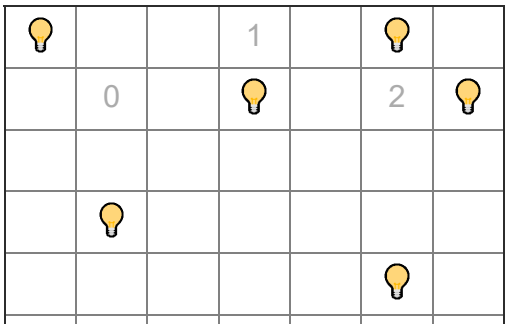
1. **The beginning of the game.** The game board only contains the tiles (mostly white, some black with or without numbers) but no light bulbs.

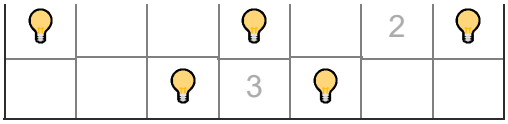


2. **Intermediate state.** The player can place a new light bulb or remove previously placed bulbs by clicking. Notice that the light only spreads along the row and column of the light bulb and does not traverse through black tiles!



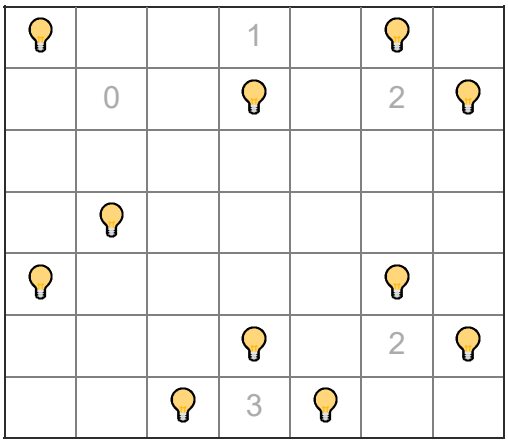
3. **The end of the game.** The player wins once every white tile is illuminated according to the rules.





Example of an incorrect solution

The solution below is **incorrect** (regardless of the fact that all white cells are actually illuminated) because two light bulbs illuminate each other!



Game implementation

The game launches on a **map selector** screen that performs the following:

- displays a short description of the game
- the map to be played can be chosen from the (pre-implemented) list of maps
- the name of the player can be entered
- the results of previous games are visible here
- the saved game (if there is one) can be continued
- the map editor can be launched from here (task for extra points)

There is also the **board** itself where the game takes place:

- the elements belonging to the selected map are displayed here (e.g. a table for the correct size but another display method can be used as well, see the help section later)
- light bulbs can be placed or removed by clicking
- the validity of the player's solution can be checked (either automatically after a stop or with the press of a button)
- the game can be saved or restarted

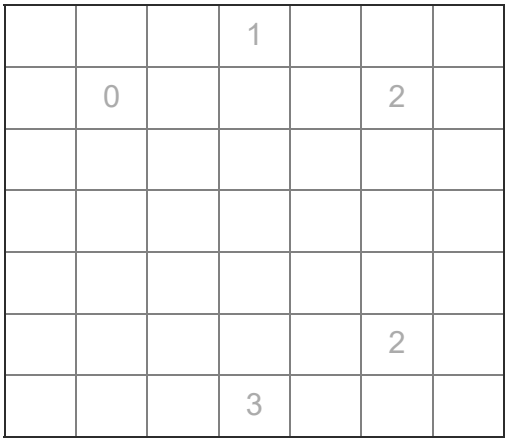
Attention! These are not separate pages, only different panels which can be hidden or shown as needed. The entire game SHOULD be on one HTML page!

Example maps

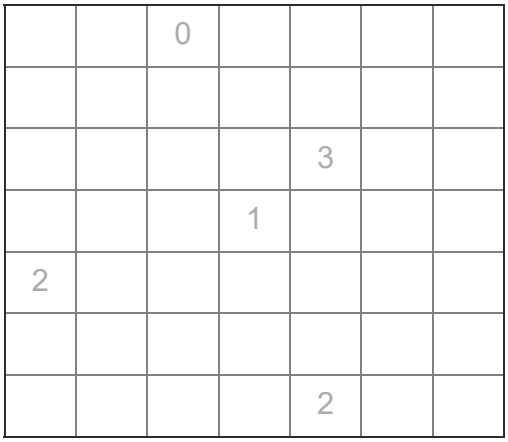
In the three tables below a starting state is provided for each level of difficulty. You may as well create your own layout but pay attention that there should be a sufficient number of black tiles and some of them should

contain numbers as well. **If you parameterize the task correctly, the puzzle will have only one solution.**

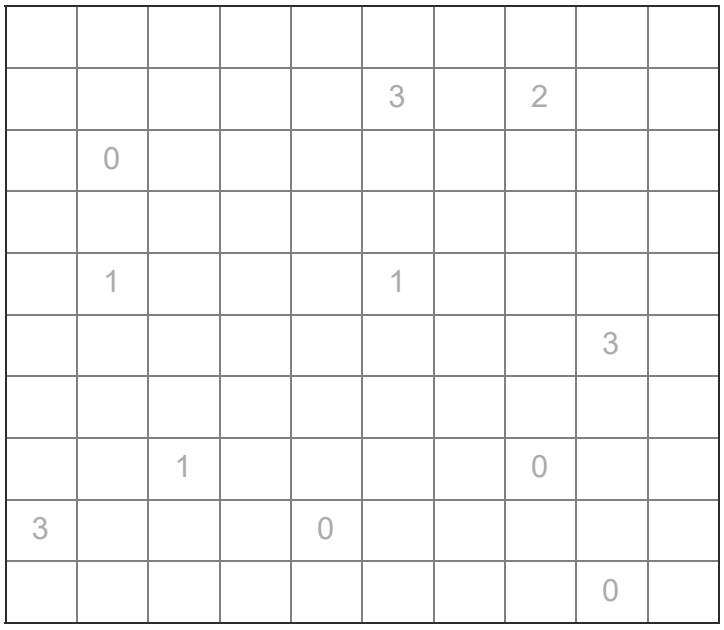
1. Easy 7x7 map:



2. Advanced 7x7 map:



3. Extreme 10x10 map:



Design

Design is important. Your submission doesn't have to be too pretty and filled with frills, but it should look nice on a screen of at least 1024×768 pixels; the grid should contain square-shaped cells. You can use

minimalistic design, custom CSS with extra graphical elements, or a CSS framework.

There's no mandatory **technology** for displaying the game: you can use `table`s, `div`s, and `canvas` freely. Criteria for function and presentation isn't set in stone, there's flexibility in the grading as long as your **game is playable well** and the tasks described above work in a recognizable way.

Help

First construct the elements that your game will need. If you are not using canvas, create the static HTML and CSS prototype of your game. Experiment and create the necessary elements:

- how do you realize the grid layout?
  - table?
  - absolutely positioned elements?
  - CSS flexbox?
  - CSS grid?
- how do you place a number or a light bulb on a tile so that it's visible well?
  - text?
  - number?
  - colour?
  - image?
  - emoji/icon?
- how do you implement the animation of the illumination?
- what does the map selector look like?

For these you do not have to program, just use your HTML and CSS skills.

Next, think about the necessary data to describe the game, and what data structure is needed for them!

- on the game board...
  - what colour are the tiles?
  - are there numbers on the black tiles?
- during the game...
  - where have we already placed light bulbs?
  - how many light bulbs surround a given black tile?

What operations are needed on those data...

- to place or remove a light bulb?
- to determine if a tile is illuminated?
- to determine if two light bulbs clash?
- to determine if the number of surrounding light bulbs correct?
- to decide if the solution is valid?

What events do you have to work with?

- event types

- source element?
- listening element?
- bubbling and delegation?

We do not see everything in a bigger task (e.g. validity of a solution) right away. You do not need to divide the HTML and CSS work to steps, you can design the whole interface in one go. However, when developing JavaScript, it's better to take small steps. Solve one thing at once, and make it work!

Grading

A total of 20 points can be obtained by solving the home assignment. There are some minimum requirements without which the assignment cannot be accepted. An additional 5 points can be obtained for extra tasks. Therefore, by solving everything you may get up to 25 points for the home assignment.

Criteria related to the JavaScript home assignment for getting your final course grade: it is required to meet all minimum requirements AND have at least 8 points (40%) after the deduction of points due to late submission. (Please consider what is easier for you: solve the minimum requirements by the deadline; or hand in a version with extra tasks completed two weeks later and still receive 8 points due to the deduction.)

Minimum requirements (must be completed, 8 pts)

- Other: The `README.md` file from the *Other requirements* section is filled with your data and included with your solution (0 pts)
- Game board: the elements of a map are shown properly (1 pt)
- Game board: light bulbs can be placed on the white tiles by clicking (1 pt)
- Game board: the placed light bulb can be removed by clicking again (1 pt)
- Game board: light bulbs cannot be placed on the black tiles (1 pt)
- Game board: the game detects (either automatically or by the click of a button) if the solution is correct (3 pts)
- Game board: the game can be restarted after solving without reloading the page (1 pt)

Basic tasks (12 pts)

- Map selector: at least three different maps can be selected and started correctly (1 pt)
- Map selector: the player's name can be entered which is shown during the game and on the scoreboard (1 pt)
- Game board: the elapsed time is always shown and updated (1 pt)
- Game board: all illuminated tiles (including the tiles containing the light bulbs) get yellow background colour (1 pt)
- Game board: the propagation of light is animated, the yellow background colour spreads from the light source after it has been placed (1 pt)
- Game board: show with a different style (e.g. green text colour) if a black tile is surrounded by the correct number of light bulbs (1 pt)

- Game board: show with a different style (e.g. red colour or icon) if two light bulbs illuminate each other (1 pt)
- Game board: the game can be interrupted and saved (1 pt)
- Map selector: the latest results can be seen - player's name, map name, time elapsed (1 pt)
- Map selector: the latest results are stored persistently after the page is closed (1 pt)
- Map selector: the saved game is shown and can be loaded properly (1 pt)
- Other: nice design (1 pt)
- **Late submission: -0.5 pts / started day!**

Extra tasks (extra 5 pts)

- Map editor: custom maps can be created with any dimension and starting tiles (3 pts)
- Map editor: the custom maps are stored persistently in localStorage (1 pt)
- Map editor: the custom maps can be reopened for editing and saved again (1 pt)

Other requirements

- The assignment should be compressed into an archive containing all necessary files AND the `README.md` file in the root folder of the program and uploaded to Canvas by the deadline (plus two week with point deduction).
- You **cannot** use any external, third-party JavaScript libraries.
- The `README.md` file has the following requirements:
  - You fill in your own data at the start of the file (marked by `< >` signs)
  - You mark all (even partially) finished subtasks with an `x` in place of the space between the square brackets `[ ]` in the file.

```
<student's name>
<student's NEPTUN code>
Web-programming - JavaScript home assignment
This solution was submitted by the stundent named above for a Web-programming assignment.
Hereby I declare that the solution is my own work. I did not copy or use solutions from a third party. I did not
 share this solution with fellow students, and I did not publish it.
According to the Academic Regulations for Students (Eötvös Loránd University Organisational and Operational Regul
ations – Volume 2, Section 74/C), a student purporting the intellectual property of others as their own [...] is
 committing a disciplinary offence.
The worst result of a disciplinary offence can be the expulsion of the student.

Minimum requirements (must be completed, 8 pts)

[ ] Other: the `README.md` file from the *Other requirements* section is filled with your data and included with
 your solution (0 pts)
[ ] Game board: the elements of a map are shown properly (1 pt)
[ ] Game board: light bulbs can be placed on the white tiles by clicking (1 pt)
[ ] Game board: the placed light bulb can be removed by clicking again (1 pt)
[ ] Game board: light bulbs cannot be placed on the black tiles (1 pt)
[ ] Game board: the game detects (either automatically or by the click of a button) if the solution is correct (3
pts)
[ ] Game board: the game can be restarted after solving without reloading the page (1 pt)

Basic tasks (12 pts)
```

```
[ ] Map selector: at least three different maps can be selected and started correctly (1 pt)
[ ] Map selector: the player's name can be entered which is shown during the game and on the scoreboard (1 pt)
[ ] Game board: the elapsed time is always shown and updated (1 pt)
[ ] Game board: all illuminated tiles (including the tiles containing the light bulbs) get yellow background colo
ur (1 pt)
[ ] Game board: the propagation of light is animated, the yellow background colour spreads from the light source
after it has been placed (1 pt)
[ ] Game board: show with a different style (e.g. green text colour) if a black tile is surrounded by the correct
number of light bulbs (1 pt)
[ ] Game board: show with a different style (e.g. red colour or icon) if two light bulbs illuminate each other (1
pt)
[ ] Game board: the game can be interrupted and saved (1 pt)
[ ] Map selector: the latest results can be seen - player's name, map name, time elapsed (1 pt)
[ ] Map selector: the latest results are stored persistently after the page is closed (1 pt)
[ ] Map selector: the saved game is shown and can be loaded properly (1 pt)
[ ] Other: nice design (1 pt)

Extra tasks (extra 5 pts)

[ ] Map editor: custom maps can be created with any dimension and starting tiles (3 pts)
[ ] Map editor: the custom maps are stored persistently in localStorage (1 pt)
[ ] Map editor: the custom maps can be reopened for editing and saved again (1 pt)
```

A submission without a properly filled `README.md` file is not eligible to be graded!