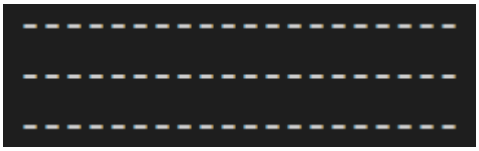
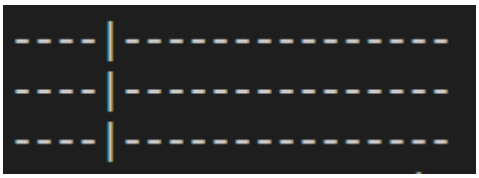
Coursework 3: Radio Tower Exercise





The provided code does the following:

- Exit the Program (Input '0')
- Display the radio towers (Input '1')
- Build a new tower (input '2')

It stores the "state" of each tower in a list.

Modification 1: Tower Destruction

Not only do we like to build towers, we also love destruction.

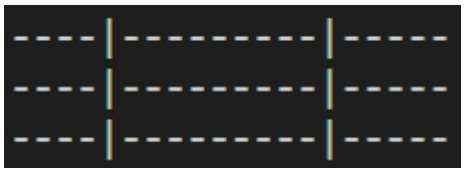
When the user inputs '2':

- Program prompts the user to input a number from 1-20
- If the corresponding location initially has a tower on it, remove it
- The change can be shown in the display function (Input '1')

Modification 2: Furthest Tower

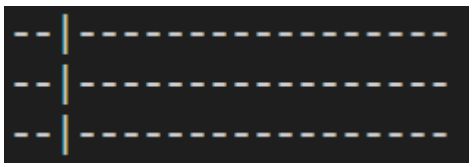
This is similar to building a new tower, but the program decides where to place the tower. The program should choose a location that is furthest to the nearest radio tower. For example:

Example 1: There are two towers at location 5 and 15:



The furthest tower will be at location 10 or location 20 (both will have a distance of 4 to the nearest existing tower). So a tower will be built on location 10 OR location 20 (doesn't matter which one).

Example 2: There is one tower at location 3:

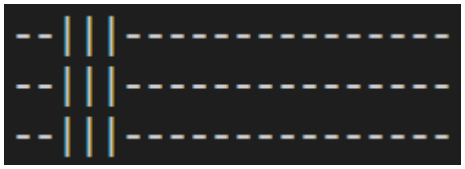


The furthest tower will be at location 20 (distance of 16), so a tower on location 20 will be built.

Modification 3: Crowdedness

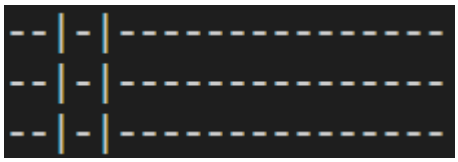
Hey, radio towers positively correlate to public health. We do not want any 2 or more radio tower to be next to each other. However, we want to do this with a minimum number of tower destruction.

Example 1: There are three towers at location 3,4,5

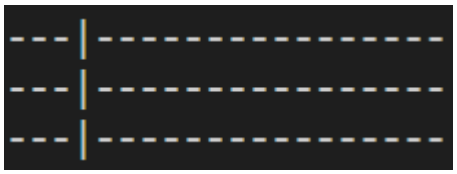


We have 3 consecutive towers.

We can remove tower 4 and it will become:



There is another incorrect way to solve this problem by removing tower 3 and tower 5:



This is wrong because it destructs 2 towers (instead of 1 tower like the correct solution)

Modification 4: Skyscraper

We can go higher! Right now, the *towers* list only contains '0' or '1'. Modify the existing functions in the following way:

- *towers* (list): Represents *height* instead of "existence"
- Build: Instead of setting a tower to be '1', add 1 to its existing height. For example if towers[3] = 3, build(3) would result in towers[3] = 4.
- Destruct: Instead of setting a tower to be '0', minus 1 from its existing height. It cannot drop below 0. For example if towers[5] = 2, destruct(5) would result in towers[5] = 1
- Furthest_tower/Crowdedness: Use the modified build/destruct function
- Display: for a tower of height n, it contains n "|" on its location .

Example: towers[4] = 1, towers[5] = 2, towers[7] = 3: