Project 6

You are going to write a complete C program which implements the following functionality:

- Your program will read a file named `input.txt`
- `input.txt` contains single line text. The text describes a tree structure. It follows the Newick format rules. But there are simplifications and/or modifications:
  - There isn't any space between the characters.
  - Each node is represented by a single character. They are separated by `comma`. Branching is described by matching parentheses.
  - The length of the text will not be greater than `100`
- Below is an example of a tree description:

  ```
  (A,(c,B,e),K,D,e,(f,(d,F)))
  ```

- This tree can be visualized as follows:

  ```
  -A
  --c
  --B
  --e
  -K
  -D
  -e
  --f
  ---d
  ---F
  ```

- Another example:

  ```
  (A,(A,A,(A),A),A)
  ```

- Visualization:

  ```
  -A
  --A
  --A
  ---A
  --A
  -A
  ```

- Another example:

  ```
  A,(A,A,(A),A),A
  ```

- Visualization:

  ```
  A
  -A
  -A
  --A
  -A
  A
  ```

- Your program will read the simplified newick formatted tree description from a file and print the visualization of the described tree to stdout.
- Core part of your implementation (parsing and printing) should utilize **recursion** otherwise you will get `0pts`. For this problem, coding without recursion is significantly easier.
- Pay attention to the structure of the output. If your program prints something slightly different or anything extra, you will loose at least `20pts`. Don't print any debug messages or greeting texts. Don't add extra spaces, newlines, commas etc. . .

Turn in:

A complete C program `<Project6.c>` which can be compiled using the following command:

```
gcc -std=c99 Project6.c -o Project6
```

If your program requires additional compile and link options, state that requirement at beginning of your source code as a comment.