CSE 102 Programming Assignment 7 (200 points. Double Assignment)

If you complete this assignment you may get 200 points. Not submitting will be considered as "not submitting 2 assignments".

DUE

January 4, 2023, 23:55

Description

- This is an individual assignment. Please do not collaborate
- If you think that this document does not clearly describes the assignment, ask questions before its too late.

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

- your program reads two files:
  - `circuit.txt`
  - `input.txt`
- According to content in `circuit.txt`, the program **dynamically** creates necessary structures for a logic circuit and evaluates the cases listed in `input.txt`.
- Your program prints the output to `stdout`. After each output there should be a newline.

`circuit.txt`

This file describes a logic circuit. Below is the file format:

```
GATE <gate_type> <gate_name>
.
.
.
GATE <gate_type> <gate_name>
GATE <gate_type> <gate_name>
CONNECTION <from_gate> <to_gate>
.
.
.
CONNECTION <from_gate> <to_gate>
CONNECTION <from_gate> <to_gate>
```

- Each line starts with a `keyword`. Possible `keywords`:

  ```
  GATE
  CONNECTION
  ```

`GATE` line describes a logic gate instance in the circuit. The first word after the `GATE` keywords is the type of the gate. Possible types:

```
INPUT
OUTPUT
AND
OR
NOT
FLIPFLOP
```

The second word is the name of the gate. Names are unique in the circuit.

`INPUT` type represents an input in the circuit. `OUTPUT` type represents an output in the circuit. `AND` type represents an `and` function. It takes 2 or more inputs. It has one output. `OR` type represents an `or` function. It takes 2 or more

inputs. It has one output. `NOT` type represents a `not` function. It takes 1 input. It has one output. `FLIPFLOP` type represents a flipflop gate It takes 1 input. It has one output.

`CONNECTION` line describes a connection between two gates. The first word after the `GATE` keywords is the name of the gate which creates the output. The second word after the `GATE` keywords is the name of the gate which receives the input.

`input.txt`

- Each line is a list of `1` and `0`. Example:

```
1011
0111
0010
1001
```

- No gaps between the bits.

Example:

- Suppose that `circuit.txt` is has the following content:

```
GATE INPUT a
GATE INPUT b
GATE INPUT c
GATE INPUT d
GATE AND and1
GATE OR or1
CONNECTION a and1
CONNECTION b and1
CONNECTION and1 or1
CONNECTION c or1
GATE NOT n1
CONNECTION d n1
GATE FLIPFLOP f1
CONNECTION n1 f1
GATE AND and2
CONNECTION or1 and2
CONNECTION f1 and2
GATE OUTPUT o
CONNECTION and2 o
```

- `input.txt` has the following content:

```
1101
1010
1110
```

- Assume that initially `former-out` of any `FLIPFLOP` is `0`.
- Any `FLIPFLOP`s should preserve the state throughout the evaluation of the whole `input.txt`.
- Each line in `input.txt` is assigned to `INPUT` gates in the order they defined in `circuit.txt`. According to the truth tables, outputs of gates are calculated. `OUTPUT` gate outputs are printed in the order they are defined in `circuit.txt`.
- For the input.txt given, the output of your program should be(There is only one `OUTPUT` gate):

```
0
1
0
```

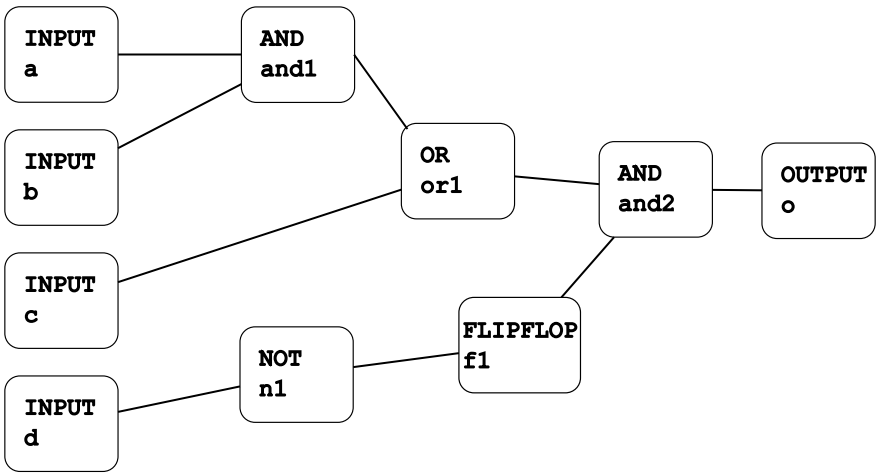

Figure 1: Example Logic Circuit

Remarks

- Each identifier is unique. Max length is `20`.
- There won't be any errors in the files.
- You have to use `dynamical memory allocation` and `struct`.
- You cannot repeatedly read `circuit.txt`. If you do, you will get `0`.
- You cannot save `circuit.txt` content to an array and repeatedly use it. If you do, you will get `0`.
- `AND` and `OR` gates can have may inputs. You have to use dynamic allocation and store the connected links.
- A `CONNECTION` is defined after the required `GATE`.
- There won't be loops in the circuit.
- Output of a gate can connect to more than one gate.
- There may be more than one `FLIPFLOP`.
- **If you don't create a linked data structure(a tree) ond use it in order to evaluate the circuit, your submission will not be accepted.**
- You cannot create arrays to store all the gate identifiers.
- You can create a pointer array which stores the address of each gate struct created. But, you cannot use this array in order to repeatedly evaluate the circuit.
- One possible struct to represent a gate can be as follows:

```
struct gate
{
    int type;               /* type of the gate*/
    char* name[20];         /* name of the gate*/
    struct gate** inputs;   /* points to a dynamically created array of pointers which store
    ↪  the addresses of input gate structs*/
    int output              /* stores the current output. this may be useful if the gate is
    ↪  connected to more than one gate.*/
    int former_out;         /* only for flipflop*/
    /*You can add other components if necessary.*/

}
```

- Write comments in your code.

- If your code does not compile you will get `0`
- Do not share your code with your classmates.
- Do not print anything other than the expected output.
- You cannot use anything which is not covered in class.
- Do not submit any of the files you used for testing.

Turn in:

- Source code of a complete C program. Name of the file should be in this format: `<full_name>_PA7.c`.
- Example: `david_gilmour_PA7.c`. Please do not use any Turkish special characters.
- You don't need to use an IDE for this assignment. Your code will be compiled and run in a command window.
- Your code will be compiled and tested on a Linux machine(Ubuntu). GCC will be used.
- Make sure you don't get compile errors when you issue this command : `gcc <full_name>_PA7.c`.
- A script will be used in order to check the correctness of your results. So, be careful not to violate the expected output format.
- Provide comments unless you are not interested in partial credit. (If I cannot easily understand your design, you may loose points.)
- You may not get full credit if your implementation contradicts with the statements in this document.

Truth Tables:

- NOT

a	out
0	1
1	0

- FLIPFLOP

a	former_out	out
0	0	0
0	1	1
1	0	1
1	1	0

Late Submission

- Late submission is NOT accepted.

Grading (Tentative)

- `Max Grade` : `200`.

All of the followings are possible deductions from `Max Grade`.

- No submission: `-200`. (be consistent in doing this and your overall grade will converge to `N/A`) (To be specific: if you miss `3` assignments you'll get `N/A`)
- Compile errors: `-200`.
- Irrelevant code: `-200`.
- Major parts are missing: `-200`.
- Unnecessarily long code: `-30`.
- Using language elements and libraries which are not allowed: `-200`.
- Not caring about the structure and efficiency: `-100`. (avoid using hard-coded values).
- Significant number of compiler warnings: `-10`.
- Not commented enough: `-10`. (Comments are in English).

- Source code encoding is not `UTF-8` and characters are not properly displayed: `-5`. (You can use 'Visual Studio Code', 'Sublime Text', 'Atom' etc... Check the character encoding of your text editor and set it to `UTF-8`).
- Fails at reading `circuit.txt`: `-150`.
- Fails at reading `input.txt`: `-150`.
- The result is wrong: `-100`.
- Output format is wrong: `-30`. (Be careful with spacing)
- Infinite loop or recursion: `-150`.
- Prints anything extra: `-30`.
- Unwanted chars and spaces in the output: `-30`.
- Submission includes files other than the expected: `-10`.
- Submission does not follow the file naming convention: `-10`.
- Sharing or inheriting code: `-400`.
- IF YOU DON'T FOLLOW THE FILE NAMING CONVENTIONS YOU WILL GET `0`.

Note: Some of these items are not independent. So, you cannot expect isolation of many of them. For example, if you cannot read input file correctly, you will fail to produce the correct output file. Partial grading is not guaranteed.