CSE 102 Homework Assignment 6

DUE

December 27, 2020, 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.

For this assignment, you are expected to write an interactive program which, according to the user input, creates dynamic responses and do simple calculations.

A compound object will be defined interactively and you program will keep track of a simple property and return a result based on the composition of the object.

Example:

Suppose that you want to calculate the total cost of a compound object. In this case your compound object is a `bicycle`. A bicycle has many parts and some of the parts are also compound objects individually. So, a hierarchical structure can be used in order to model such a compound object. You can visualize it as a tree. Nodes are the parts and the children of a particular node are the parts which make up the part associated with that node.

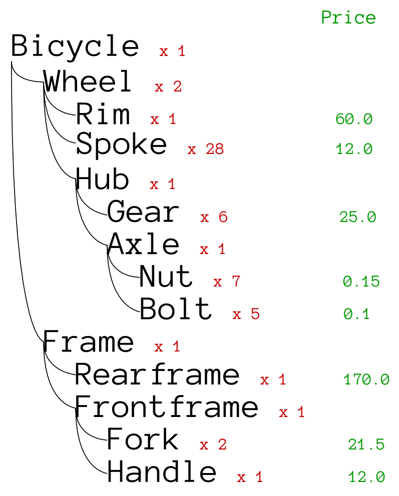

Figure 1: Composition of a bicycle

`Figure 1` shows parts of a bicycle. A bicycle can be described as an object which consists of 2 wheels and a frame. We can create a more detailed description by including the description of the individual parts, `wheel` and `frame`. A description includes number of the parts, part name and, if no further description is needed, prices of the part or parts.

Your program does not know anything about the compound object. The expectation is; either there is a compound object which can be specified by its parts or there is a simple object which can be specified by its `property`.

For this example, the `propery` is the price of the object. Your program will ask questions to the user and eventually calculate the total price of the bicycle described by the user.

Below is a simple dialog which the user describes the object 'bicycle' to the program: (Comments are not parts of the output)

```
> Name the object:
> Bicycle
> Define Bicycle?:
> 2 Wheel 1 Frame                                   Comment: A bicycle consists of 2 Wheels and 1 Frame.
> Define Wheel in Bicycle?:
> 1 Rim 28 Spoke 1 Hub
> Define Rim in Wheel?:
> 60                                               Comment: The price of a Rim is 60.
> Define Spoke in Wheel?:
> 12.0000000000000
> Define Hub in Wheel?:
> 6 Gear 1 Axle
> Define Gear in Hub?:
> 25.0
> Define Axle in Hub?:
> 7 Nut 5 Bolt
> Define Nut in Axle?:
> 0.15
> Define Bolt in Axle?:
> 0.1
> Define Frame in Bicycle?:
> 1 Rearframe 1 Frontframe
> Define Rearframe in Frame?:
> 170.0
> Define Frontframe in Frame?
> 1 Fork 1 Handle
> Define Fork in Frontframe?:
> 21.5
> Define Handle in Frontframe?:
> 12
> Total cost of Bicycle is 1320.1
```

Remarks

- Do not assume anything about the depth of the tree. Don't try to store the whole tree in the memory. This may not be possible.
- You don't have to do error checking for the input.
- You will construct the required data structure on-the-fly.
- There may be spaces in between the terms of the user input. Below is a valid input:

  ```
  1       Rearframe      1      Frontframe
  ```

- Do not submit your code without testing it with several different scenarios. `Bicycle` and its structure are presented as an example only. Your program should run if the user tries to describe another object. For example: A car engine, a meal, etc…
- You can use c structs, unions, arrays, c strings, pointers, recursion.
- Write comments in your code.
- Do not print anything other than the expected output.
- Do not submit any of the files you used for testing.
- Do not submit your output file.

Turn in:

- Source code of a complete C program. Name of the file should be in this format: `<full_name>_<id>.c`.
- Example: `gokhan_kaya_000000.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>_<id>.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.

Late Submission

- Late submission is NOT accepted.

Grading (Tentative)

- `Max Grade` : `100`.

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

- `#define HARD_CODED_VALUES -10`.
- No submission: `-100`. (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: `-100`.
- Irrelevant code: `-100`.
- Major parts are missing: `-100`.
- Unnecessarily long code: `-30`.
- Using language elements and libraries which are not allowed: `-100`.
- Not caring about the structure and efficiency: `-30`. (avoid using hard-coded values).
- Significant number of compiler warnings: `-10`.
- Not commented enough: `-5`. (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 reading an input with spaces: `-20`.
- Fails parsing part names. `-30`.
- Fails parsing part values. `-30`.
- Fails parsing part counts. `-30`.
- Fails during file write: `-30`.
- Wrong hierarchy at the output: `-30`.
- Prints wrong names: `-30`.
- Output format is wrong: `-30`.
- Infinite loop: `-90`.
- Prints anything extra: `-30`.
- Unwanted chars and spaces in `output.txt`: `-30`.
- Submission includes files other than the expected: `-10`.
- Submission does not follow the file naming convention: `-10`.
- Sharing or inheriting code: `-200`.
- 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.