MIDDLE EAST TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

CENG 111

Fall 2024

Take-Home Exam 2

REGULATIONS

Due date: 23:59, 8 December 2024, Sunday *(Not subject to postpone)*

Submission: Electronically. You should save your program source code as a text file named `the2.py`. Check the announcement on the ODTUCLASS course page for the submission procedure.

Team: There is **no** teaming up. This is an EXAM.

Cheating: Source(s) and Receiver(s) will receive zero and be subject to disciplinary action.

INTRODUCTION: SOCIAL TAXING SYSTEM

"It's a simple recipe. Add politics to economics, you get tax law."

by Chris Wildt.

A "social taxing system" is all about making sure everyone gets a fair share and that we take care of each other. It takes into account what each taxpayer is going through and makes sure that everyone pays a fair amount. A social tax system is different from other tax models because it takes into account many different aspects. These include e.g. how much money you make, who you are responsible for at home, how long you have been paying taxes, and the cost of living in your area. This way, those who have more money pay more tax, while those who have less or who have to look after other people or who live in expensive places get help. This is a fair(er) way of tax collection. It can help make the economy stronger and help people to keep more of their money when they need it most. This makes people's lives better and helps reduce poverty.

PROBLEM & SPECIFICATIONS

In this THE, your task is to write Python code that calculates the tax for a person. The input is going to be provided as a dictionary (`dict`) with the following syntax:

- **INCOME**
  - **Description**: Specifies the income category of the individual. Determines base tax rate.
  - **Properties**: `low`, `middle`, `high`.
  - **Example Entry**: {`...  "INCOME":  "middle" ...`}
- **MARITAL_STATUS**
  - **Description**: Specifies the marital status of the individual. Impacts tax deductions or credits.
  - **Properties**: `single`, `married`, `single_parent`.
  - **Example Entry**: {`...  "MARITAL_STATUS":  "married" ...`}
- **DEPENDENTS**
  - **Description**: Details about children and elderly dependents supported by the individual.
  - **Sub-properties**:
    - **CHILD**: List of child ages. If no child: empty list.
    - **SPECIAL_NEEDS**: Boolean (`True` if there is at least one child that has special needs.
    - **ELDERLY_CARE**: Boolean (`True` if caring for an elderly relative).
  - **Example Entries**:
    - {`...  "CHILD":  [12,13,19] ...`}
    - {`...  "SPECIAL_NEEDS":  True ...`}
    - {`...  "ELDERLY_CARE":  True ...`}
- **TAXPAYER_DURATION**
  - **Description**: Specifies how long the individual has been a taxpayer, which may impact deductions.
  - **Properties**: `new`, `regular`, `long_term`.
  - **Example Entry**: {`...  "TAXPAYER_DURATION":  "long_term" ...`}
- **CITY_CATEGORY**
  - **Description**: Indicates the individual's primary residence, influencing tax reliefs for living costs.
  - **Properties**: `urban`, `suburban`, `rural`.
  - **Example Entry**: {`...  "CITY_CATEGORY":  "suburban" ...`}
- **DEDUCTIONS**

  - **Description**: List of additional deductions based on various qualifying factors.
  - **Sub-properties**:
    * **EDUCATION**: Boolean (`True` if education deduction applies).
    * **HEALTHCARE**: Boolean (`True` if any healthcare-related deductions apply).
    * **GREEN_INITIATIVES**: Boolean (`True` if using renewable energy or owning an electric vehicle).
  - **Example Entries**:
    * {... "EDUCATION": True ...}
    * {... "HEALTHCARE": False ...}
    * {... "GREEN_INITIATIVES": True ...}

- **PROPERTY_STATUS**
  - **Description**: Specifies whether the individual owns property or is renting.
  - **Properties**: `owns`, `rents`.
  - **Example Entry**: {... "PROPERTY_STATUS": "owns" ...}

Tax Calculation

The following rules describe how annual tax calculations are structured based on the dictionary entries provided:

- **Base Tax Rate**:
  - **INCOME**: `low` - 10%, `middle` - 20%, `high` - 30%.
- **Marital and Family Deductions** (from the base tax):
  - If **SINGLE**: No family-related deductions.
  - If **MARRIED**: Deduct €500 for spouse; additional €300 per **CHILD**.
  - If **SINGLE_PARENT**: Deduct €600 per **CHILD**.
- **Dependent Adjustments**:
  - For each child with age below 18: Additional €200 deduction.
  - If **SPECIAL_NEEDS**: Additional €1,000 deduction.
  - If **ELDERLY_CARE**: Additional €800 deduction.
- **Residence-Based Deduction**:
  - **CITY_CATEGORY**
    * `urban`: No residence deduction.
    * `suburban`: Deduct €200.
    * `rural`: Deduct €400.
- **Additional Deduction Criteria**:
  - **EDUCATION**: €500 deduction if `True`.

  - **HEALTHCARE**: €750 deduction if `True`.
  - **GREEN_INITIATIVES**: €300 deduction if `True`.

- **Property Adjustment**:
  - **PROPERTY_STATUS**:
    * `owns`: No additional deduction.
    * `rents`: Deduct €300.

- **Duration-Based Deduction**:
  - **TAXPAYER_DURATION**
    * `new`: No additional deductions.
    * `regular`: Deduct 5% from total tax (total tax is the tax amount left over after all other deductions (other than this Duration-Based Deduction) are made).
    * `long_term`: Deduct 10% from total tax.

- **Minimal value for tax**: The final tax amount cannot be negative. Any such, will be set to zero.

RESTRICTIONS & GRADING

- Input is a line containing a dictionary as explained above (containing all the properties of the tax payer), followed by a second line (a `float`) that is the annual income of the tax payer.

  You can read each input line with:
  `eval(input())`

- The output is the final tax amount, printed as follows:
  `print("%.2f" % final_tax_amount)`

- You are strictly forbidden from using Python's repetitive statements (`for` and `while`).

- A set of arbitrarily selected students will be subject to oral examination about their solutions. The details will be announced on ODTUclass later.

- Your program will be graded through an automated process and therefore, any violation of the specifications will lead to errors (and reduction of points) in automated evaluation. You should not print anything other than the result.

- Your solutions will not be tested with incorrect/erroneous/incomplete inputs. This implies that all keys of the dictionary will be present.

- Your program will be tested with multiple data (a distinct run for each data). Any program that performs only 30% and below will enter a glass-box test (eye inspection by the grader TA). The TA will judge an overall THE2 grade in the range of [0, 30].

- A program based on randomness will be graded zero.

- The glass-box test grade is not open to discussion nor explanation.

SAMPLE RUN

Example Input

The following sample input is displayed on multiple lines for the sake of visualization. You should place the whole dictionary on a single line to be able to provide the dictionary as input (via copy-paste or in a script):

```
  {"INCOME": "middle", "MARITAL_STATUS": "married", "CHILD": [12,13,19],
   "SPECIAL_NEEDS": False, "ELDERLY_CARE": True,
  "TAXPAYER_DURATION": "regular", "CITY_CATEGORY": "suburban",
  "EDUCATION": True, "HEALTHCARE": False, "GREEN_INITIATIVES": True,
  "PROPERTY_STATUS": "rents"}
36000.0
```

Example Output

```
3135.00
```