ZADANIE

Napisz oprogramowanie w C#, Python bądź Go (wybierz jeden), które domyślnie przyjmuje na wejściu plik input.xml (plik w formacie XML), oraz domyślnie zwraca na wyjściu plik output.json (plik w formacie JSON). Plik wykonywalny programu nie powinien wymagać żadnych parametrów podawanych w wierszu poleceń.
Interpretowane z pliku XML maja być tylko pola opisane następującymi słowami kluczowymi: object, obj_name, field, name, type, value.

object - definiuje zakres obiektu
obj_name - definiuje nazwę obiektu
field - określa zakres pola obiektu
name - określa nazwę dla pola obiektu
type - określa typ pola obiektu
value - określa wartość pola obiektu

Dla każdego obiektu, aby uznać go za poprawny, powinna być przypisana nazwa, oraz przynajmniej jedno pole. Nazwy obiektów są wartościami unikalnymi. Dla każdego pola, aby je uznać za poprawne, powinna być zdefiniowana nazwa, typ oraz wartość. Dozwolone są dwa typy danych 'int' oraz 'string'.
Na etapie konwersji z xml do json należy zignorować (pominąć) wszystkie:

- pola obiektów które zawierają niedozwolone typy danych,
- obiekty które posiadają niewspierane słowa kluczowe,
- pola które posiadają niewspierane słowa kluczowe,
- obiekty które są niepoprawne,
- wszystkie pola które są niepoprawne,
- wszystko co zawiera nie drukowalne znaki (non-printable characters).

Zadanie należny dostarczyć w formie, która umożliwia jego uruchomienie bez wymaganych dodatkowych kroków (skompilowany plik binarny, bądź skrypt Python, wraz ze wszystkimi zależnościami jak np. dodatkowe biblioteki). Wszystkie pliki wymagane do uruchomienia oprogramowania powinny znajdować się w folderze o nazwie odpowiadającej formatowi imie.nazwisko autora oraz skompresowane "zipem", pliki źródłowe zadania (bądź cały projekt) powinny znajdować się w podfolderze 'source'. Jeżeli się zastanawiasz jak nazwać projekt, luźną propozycją z naszej strony jest 'swi' (ale nie jest to wymagana nazwa). Jeżeli chcesz nam coś przekazać, najlepiej umieścić wiadomość w pliku readme.txt (np. wersję Python której używałeś). Po przygotowaniu pliku .zip wrzuć go np. na dysk google, one drive, drobpbox, github (jako release), czy jakiekolwiek inne medium do dzielenia się plikami, pamiętaj aby plik był udostępniony, oraz wyślij do nas link do pobrania pliku. Nie wysyłaj pliku mailem, ponieważ wiadomość do nas nie dojdzie (a np. gmail nawet Ci na to nie pozwoli).

Przykładowy plik input.xml

```
<object>
  <obj_name>hero_profile 1</obj_name>
  <field>
    <name>favorite_fruit</name>
    <type>string</type>
    <value>Truskawa</value>
  </field>
  <field>
    <name>hero_name</name>
    <type>string</type>
    <value>batman</value>
  </field>
  <field>
    <name>age</name>
    <type>int</type>
    <value>32</value>
  </field>
</object>
<object>
  <obj_name>hero_profile Z</obj_name>
  <field>
    <name>favorite_fruit</name>
    <type>string</type>
    <value>Kokos</value>
  </field>
  <field>
    <name>hero_name</name>
    <type>string</type>
    <value>Tomy Lee Jonse w sciganym</value>
  </field>
  <field>
    <name>age</name>
    <type>int</type>
    <value>47</value>
  </field>
</object>
<object>
  <obj_name>kolorowe kredki 1</obj_name>
  <field>
    <name>kolor</name>
    <type>string</type>
    <value>fuksja</value>
```

```
  </field>
</object>
<object>
  <obj_name></obj_name>
</object>
<object>
  <obj_name>kolorowe kredki B</obj_name>
  <field>
    <name>kolor</name>
    <type>string</name>
  </field>
  <lolfield>lol"</lolfield>
  <field>
    <name>kolor</name>
    <type>int</type>
    <value>42</value>
  </field>
</object>
<object>
  <obj_name>Buty  Plytowe Ogolnego Gniewu i Zniszczenia </obj_name>
  <field>
    <name>Armor</name>
    <type>int</type>
    <value>12</value>
  </field>
</object>
```

Oczekiwany plik result.json (powinien powstać na podstawie przykładowego pliku input.xml podanego w zadaniu)

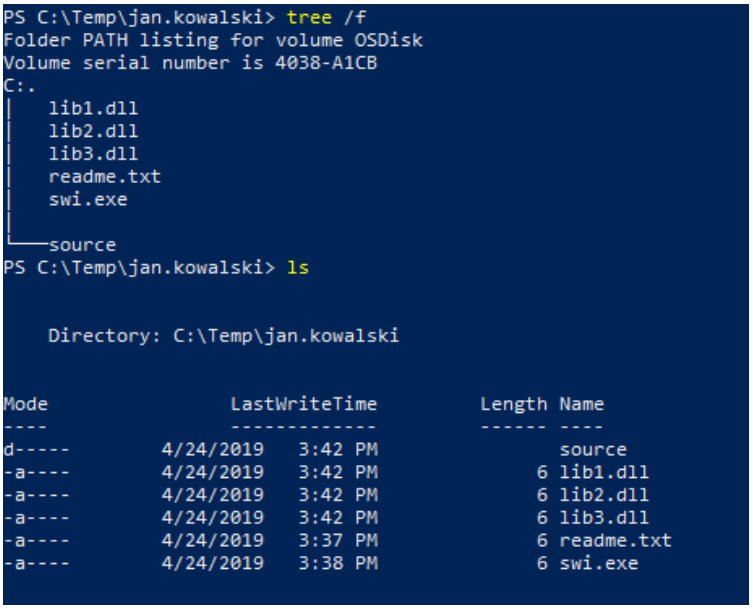
```
{
  "hero_profile 1": {
    "favorite_fruit": "Truskawa",
    "hero_name": "batman",
    "age": 32
  },
  "hero_profile Z": {
    "favorite_fruit": "Kokos",
    "hero_name": "Tomy Lee Jonse w sciganym",
    "age": 47
  },
  "kolorowe kredki 1": {
    "kolor": "fuksja"
```

```
  },
  "kolorowe kredki B": {
    "kolor": 42
  },
  "Buty  Plytowe Ogolnego Gniewu i Zniszczenia ": {
    "Armor" : 12
  }
}
```

Osoba weryfikująca zadanie wykona następujące czynności:

1. Rozpakuje plik imie.nazwisko.zip
2. Sprawdzi czy jest pod katalog 'source'
3. Sprawdzi czy jest plik readme.txt, jeżeli plik istnieje to z ciekawości go otworzy (w takim wypadku lepiej żeby nie był pusty).
4. Wklei do katalogu weryfikacyjny plik "input.xml" (oczywiście inny niż przykładowy plik z zadania)
5. Uruchomi program*, sprawdzi plik wyjściowy (krok ma na celu weryfikacje czy oprogramowanie działa zgodnie z założeniami)
6. Wklei do katalogu drugi plik weryfikacyjny "input.xml" (mający na celu sprawdzenie czy oprogramowanie się 'wysypie')
7. Uruchomi program*, sprawdzi plik wyjściowy
   * program będzie uruchamiany z poziomu powłoki systemu (bash/powershell) (Windows Server, bądź Linux Debian 9).

Przykładowa struktura katalogu z wynikami (pamiętaj aby katalog spakować .zip oraz do sprawdzenia przesłać wyłącznie plik .zip)



Jeżeli czegoś nie jesteś pewien/pewna, nie panikuj, sprawdź jeszcze raz instrukcje, jeżeli w instrukcji nie ma interesujących Cie informacji, zrób to co podpowiada Ci zdrowy rozsadek.
Jedynym ograniczeniem jakie nakładamy na realizację zadania jest język programowania (możliwości wymienione na początku zadania), najważniejsze dla nas jest aby oprogramowanie działało. Jeżeli pisanie zadania tak Cię rozbawi że będziesz myślał "co jeszcze by tu dopisać", śmiało możesz dopisać unit testy, testy End2End bądź stworzyć więcej weryfikacyjnych plików wejściowych xml. Ale proszę nie próbuj pisać UI czy instalatora do oprogramowania.

Jeżeli weryfikacja oprogramowania zakończy się sukcesem, będziemy się kontaktować w sprawie umówienia rozmowy w biurze Solarwinds w Krakowie (kompleks Bonarka 4 Business). Porozmawiamy wtedy z Tobą na temat Twojego programu, oraz zostaniesz poproszona(y) o drobne rozszerzenie funkcjonalności (więc pamiętaj żeby w pliku imie.nazwisko.zip dostarczyć źródła które się kompilują).