1. Stwórz przykładowy dokument w standardzie XHTML 1.0 w wersji strict. Plik powinien mieć rozszerzenie .html. Przećwicz użycie znacznika `<br>`. Dokonaj walidacji dokumentu za pomocą walidatora: http://validator.w3.org/. Popraw błędy.
2. Sprawdź, czy przykładowy dokument `book-bad.xml` jest dobrze uformowany. W tym celu:
    a. Pobierz plik `book-bad.xml` i zapisz do dowolnego katalogu.
    b. Uruchom środowisko Netbeans.
    c. Z menu „File" wybierz „Open file" i otwórz pobrany plik.
    d. Z menu „Run" wybierz „Check XML"

    Popraw błędy niepoprawnego sformułowania. Nie zamykaj Netbeans'a po skończeniu ćwiczenia. Będzie również potrzebne w kolejnym.
3. Plik `book.dtd` zawiera definicję typu dokumentu (DTD) dla pliku `book-bad.xml`. Umieść w pliku `book-bad.xml` deklarację DOCTYPE odwołującą się do pliku `book.dtd` (deklaracja prywatna). Dokonaj walidacji pliku `book-bad.xml`. W tym celu:
    a. Pobierz plik `book.xml` i zapisz do tego samego katalogu w którym znajduje się plik `book-bad.xml`.
    b. Zakładając, że masz środowisko Netbeans uruchomione i otwarty plik `book-bad.xml`, wystarczy, że teraz wybierzesz z menu „Run" polecenie „Validate XML".

    Popraw błędy tak, aby dokument XML był zgodny z DTD. Ostateczną wersję zapisz jako `book.xml`.
4. Zmodyfikuj DTD dla książki tak, aby możliwe było używanie znaczników <b> i <i> wewnątrz akapitów tekstowych. Wstaw takie znaczniki do przykładowego dokumentu i przetestuj zgodność z DTD. Sprawdź czy możliwe jest (za pomocą funkcji walidacji dokumentu XML) zagnieżdżanie elementów <b> i <i>, np.:
    ```
    <para>Przykładowy <b>akapit <i>tekstowy</i></b>.</para>
    ```
5. Wyświetl dokument `book.xml` w przeglądarce dołączając do niego styl CSS. Wskazanie na styl wymaga dodania poniższej instrukcji sterującej za prologiem dokumentu XML:
    ```
    <?xml-stylesheet type="text/css" href="styl.css"?>
    ```
    Ustaw następujące własności w dokumencie CSS:
    - element `<book>`:
        - `display:block`
        - czcionka times new roman
    - element `<title>`, dziecko elementu `<book>`
        - `display:block`
        - wariant czcionki: small-caps
        - czcionka pogrubiona
        - tekst wycentrowany
    - element `<title>`, dziecko elementu `<chapter>`
        - `display:block`
        - tekst wycentrowany
    - element `<title>`, dziecko elementu `<section>`
        - `display:block`
        - czcionka pochylona
        - tekst wyrównany do lewej
    - element `<chapter>`
        - display:block;

    - tekst wyjustowany
- element `<sekcja>`
    - `display:block;`
- element `<para>` z wartością atrybutu `type` równą `quote`
    - `display:block;`
    - tekst wycentrowany
    - czcionka pochylona
- element `<para>` z wartością atrybutu `type` równą `note`
    - `display:block;`
    - tekst wyrównany do lewej
    - czcionka monospace
- element `<para>` z wartością atrybutu `type` równą `simple`
    - `display:block;`
    - wcięcie pierwszego wiersza 3em
- element <important>
    - czcionka czerwona, pogrubiona
- element `<b>`
    - czcionka pogrubiona
- element `<i>`
    - czcionka pochylona

6. Zaprojektuj język znacznikowy (oparty o XML) do reprezentacji książki adresowej. Książka powinna przechowywać informacje o jej właścicielu (imię, nazwisko, email). Każda pozycja książki adresowej powinna przechowywać: nazwisko, adres email, nr tel., adres strony WWW i opis. Opis może zawierać podzbiór znaczników formatujących XHTML: `<b>`, `<i>`, `<tt>` (znaczniki mogą być zagnieżdżone).
7. Rozbuduj zaprojektowany język o możliwość definiowania grup osób. Grupy powinny móc tworzyć hierarchię. Dane powinny więc być zorganizowane na podobnej zasadzie jak struktura plików (grupa to odpowiednik katalogu, a pojedyncza osoba to odpowiednik pliku).
8. Stwórz arkusz styli css formatujący książkę telefoniczną napisaną zgodnie z wariantem języka znaczników z 6 zadania (nie trzeba obsługiwać grup).