Podstawowy warsztat informatyka — lista 5

Zadanie 1 (1 punkt). Ściągnij z systemu skos na dysk plik `przyklad.tex`. Ściągnięty plik skompiluj jeden raz poleceniem `pdflatex` i otwórz plik wynikowy w przeglądarce plików pdf. Następnie utwórz kopię pobranego pliku o nazwie `sprawozdanie.tex` i skompiluj ją dwukrotnie. Czy widzisz jakieś różnice w otrzymanych pdfach?

W pliku `sprawozdanie.tex` usuń wszystko pomiędzy `\begin{document}` a `\end{document}`. Spróbuj testowo wpisać tam własny tekst, a następnie skompilować ten plik. Upewnij się, że edytujesz go we właściwym kodowaniu (domyślnie UTF-8).

Zadanie 2 (2 punkty). Zmień dane osobowe w pliku `sprawozdanie.tex` na swoje oraz dodaj polecenie `\maketitle` zaraz po `\begin{document}`. Następnie utwórz pięć rozdziałów (ang. section), wzorując się na pliku przykładowym. Następnie:

- W pierwszym rozdziale wytłumacz własnymi słowami, co zwraca polecenie `uname -a`. Opis powinien być zrozumiały dla osób, które mają mgliste pojęcie o systemie Linux.
- Przed napisaniem drugiego rozdziału dowiedz się, co to jest czas uniksowy (zwany również czasem POSIX). Następnie opisz, na czym polega problem roku 2038. Do swoich danych osobowych (umieszczonych wewnątrz `\author{}`) dopisz, po przecinku, swoją datę urodzenia zapisaną w czasie uniksowym (możesz wybrać dowolną godzinę tego dnia).
- W kolejnym rozdziale opisz własnymi słowami, z czego wynikała różnica pomiędzy plikiem przykładowym skompilowanym raz, a skompilowanym dwa razy.
- Zapisz wzór z zadania 227 z whitebook'a:

Zadanie 227. $\bigcup_{n=0}^{\infty} \bigcap_{k=n}^{\infty} A_k \subseteq \bigcap_{n=0}^{\infty} \bigcup_{k=n}^{\infty} A_k$

Zadanie 3 (2 punkty). Plik `tweets.txt` zawiera teksty różnych tweetów po jednym w każdej linijce. Przetwórz ten plik odpowiednimi komendami w linii poleceń (używając rozszerzonych wyrażeń regularnych) tak aby wypisać:

1. wszystkie wzmianki (tzn. napisy wyglądające z grubsza tak: @piotrek) niezależnie czy w środku, na końcu czy na początku linii. Nie pomyl ich z innymi napisami ze znakiem `@`
2. wszystkie adresy e-mail (tzn. napisy wyglądające z grubsza tak: piotrek@cs.uni.wroc.pl) pojawiające się w tekście tweetów, niezależnie czy w środku, na końcu czy na początku linii. Nie pomyl ich z innymi napisami ze znakiem `@`
3. wszystkie słowa (złożone z co najmniej 3 liter) występujące w tekście jakiegoś tweeta zaraz po słowie `Trump`.
4. wszystkie adresy internetowe `http`, ale nie `https`.
5. liczbę adresow internetowych z domeny `t.co`.

Wypisuj dokładnie to czego szukasz, a nie całe linie. Przyjmij jakieś rozsądne definicje poszukiwanych pojęć, nie musisz rozpatrywać wszystkich możliwych przypadków brzegowych - pokaż, że rozumiesz czym są wyrażenia regularne, jak ich używać oraz że potrafisz znaleźć opis tego czego w danej chwili nie pamiętasz.

Zadanie 4 (3 punkty). To zadanie nie wymaga wcześniejszej znajomości języka Prolog ani algorytmu quicksort. Wystarczy wiedzieć, że na pracowniach jest zainstalowany interpreter Prologa, który można odpalić poleceniem `swipl`, a kod w Prologu to ciąg klauzul. Klauzule są typowo postaci:

```
wniosek :- przeslanka1, ..., przeslankan.
```

Taka linia odpowiada implikacji $przeslanka1 \wedge \ldots \wedge przeslankan \Rightarrow wniosek$.

Utwórz fork repozytorium `https://github.com/iiuwr21/quicksort17.git` i sklonuj go na swój komputer. Reprezentuje ono historię pewnej wadliwej implementacji algorytmu quicksort w języku Prolog. Twoim zadaniem będzie poprawić tę implementację. W tym celu:

1. Przejrzyj historię commitów. Znajdź (na podstawie opisów) taki, który jest jasnym oszustwem polegającym na usunięciu testu, który nie działa.

2. Ściągnij wcześniejszą wersję i sprawdź, czy rzeczywiście odpalenie programu z testami (poleceniem opisanym w README.md) zwraca informację, że jeden z testów nie jest spełniony.
   Uwaga: Ten test jest zrandomizowany i z małym prawdopodobieństwem może być spełniony, więc lepiej uruchomić go dwa razy. Stąd wziął się cały problem – w czasie edycji, która wprowadziła błąd, akurat przypadkiem test przeszedł.

3. Pobieraj coraz to wcześniejsze wersje tego programu aż dojdziesz do takiej, gdzie oba testy są spełnione. Wywnioskuj (na podstawie opisu kolejnego commita), co zostało popsute i jak to naprawić.

4. Ściągnij jeszcze raz najnowszy commit i napraw program. Przywróć wcześniejszą wersję pliku z testami aby upewnić się, że wszystko działa.

5. Zatwierdź zmiany, zrób git push i stwórz pull request dotyczący dodania Twojego komita do gałęzi master w repozytorium `https://github.com/iiuwr21/quicksort17.git`