Podstawowy warsztat informatyka — lista 8

Zadanie 1. (1 punkt) Wejdź na stronę:
`https://github.com/microsoft/Bringing-Old-Photos-Back-to-Life`
Obejrzyj otwarte i zamknięte pull requesty. Co robi microsoft-cla bot? Znajdź „Network graph": jak są na nim zaznaczone pull requesty?

Na koniec obejrzyj plik README.md. Podobnego pliku będziemy oczekiwać w każdym projekcie.

Zadanie 2. (1 punkt) 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 $przeslanka1 \wedge \ldots \wedge przeslankan \Rightarrow wniosek$.

Ściągnij repozytorium `https://github.com/j-michaliszyn/quicksort17/`. Reprezentuje ono historię pewnej wadliwej implementacji algorytmu quicksort w języku Prolog. Twoim zadaniem będzie poprawić tę implementację (nie znając Prologa ani quicksorta). 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.

Zadanie 3. (2 punkty) Jak już wiemy, jak naprawić błąd, to warto się to wiedzą podzielić i naprawić kod w oryginalnym repozytorium. Oczywiście nie chcę dawać wszystkim studentom dostępu do pisania w moim repozytorium. Właśnie dla takich sytuacji jest zaprojektowany mechanizm Pull Request.

1. Utworzyć fork wspomnianego repozytorium i sklonować go na swój komputer, a następnie poprawić pliki w sposób opisany w poprzednim zadaniu.
2. Stworzyć nowy commit oraz zrobić git push.
3. Stworzyć pull request dotyczący dodania swojego commita do jedynej gałęzi w oryginalnym repozytorium.

Zadanie 4. (3 punkty) To zadanie należy wykonywać w parach. Jeśli nie masz pary, to zrób załóż sobie drugie konto na githubie i zrób wszystko samodzielnie...

Polecenia są podzielone na dwie osoby: A i E.

E Utwórz jakieś niepuste repozytorium na githubie.

A Zrób forka tego repozytorium, wprowadź jakieś zmiany i przygotuj pull requesta.

E Skomentuj te zmiany i poproś o poprawki (Request changes).

A Wprowadź poprawki, zrób odpowiedni `git commit --amend`, następnie spuszuj zmiany[1].

E Zaakceptuj pull requesta, a następnie go scal używając „Squash and merge".

Następnie zamieńcie się rolami i powtórzcie eksperyment.

[1] To w zasadzie jest jedyna *zwykła* sytuacja, w której wolno użyć git push -f — gdy puszujemy do swojego roboczego forka.

Zadania bonusowe

Zadanie 5. (2* punkty) Dowiedz się, jak działają hooki w gicie. Stwórz hook (w dowolnym języku programowania, którego interpreter lub kompilator jest na pracowni), który sprawdza, czy podany opis commita ma co najwyżej 50 znaków w pierwszej linii. Jeśli nie, commit powinien zostać odrzucony ze stosowną informacją zwrotną. Przygotuj repozytorium do testowania i przeprowadź dwa testy sprawdzające działanie hooka: jeden z commitem z dobrym opisem, a drugi ze złym.

Zadanie 6. (2* punkty) Dowiedz się, jak działa `git bisect`[2]. Przeprowadź eksperyment dotyczący `git bisect`. W tym celu stwórz skrypt (w dowolnym języku programowania), który:

- Utworzy puste repozytorium oraz doda do niego plik `log` o treści „Kolejne czasy:".
- Wylosuje liczbę i między 1 a 1024.
- Wykona $i - 1$ razy operację `date >> log` a po niej `git commit -a -m "Dodana data"`.
- Wykona operację[3] `date > log` a po niej `git commit -a -m "Dodana data"`.
- Wykona $1024 - i$ razy operację `date >> log` a po niej `git commit -a -m "Dodana data"`.

Następnie wykorzystaj `git bisect` do znalezienia commita, w którym nastąpiło nadpisanie.

Uwaga: Do tego zadania trzeba zainstalować qemu i pobrać pół gigabajta danych.

Zadanie 7. (2* punkty) Użyj programu wget do ściągnięcia do katalogu `/var/tmp` pliku `ii.uni.wroc.pl/static/obraz.7z` zawierającego obraz pewnej instalacji Debiana. Ściągnięty plik rozpakuj i nazwij

`debian-twoj_numer_indeksu.qcow2`

Uruchom pobrany obraz poleceniem

`qemu-system-i386 -hda /var/tmp/debian-twoj_numer_indeksu.qcow2`

lub podobnym. Po kilku chwilach powinien powitać cię tekstowy ekran logowania. Niestety, nie znasz hasła roota. Co za pech.

Znajdź w internecie informację o tym, jak zmienić hasło roota i zmień je. To dużo łatwiejsze, niż by się mogło wydawać. Następnie uruchom system ponownie i zaloguj się na konto roota.

Ten system jest trochę nieaktualny. Dowiedz się, jak go zaktualizować programem `apt-get`. Następnie zainstaluj program `git` i go uruchom.

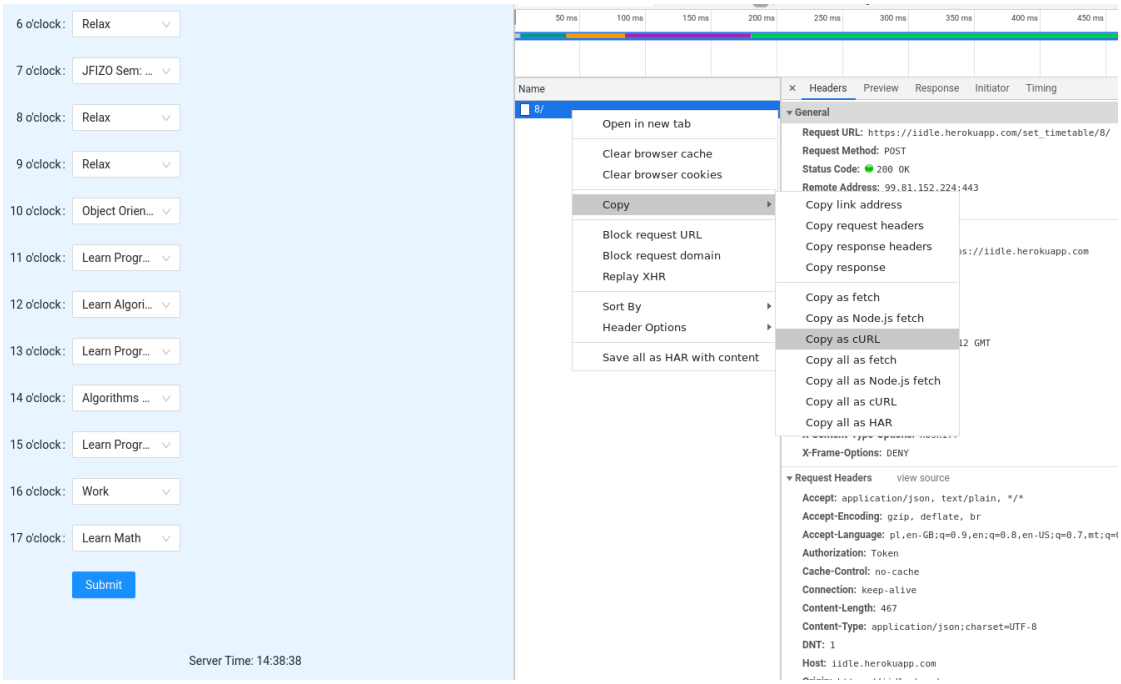
Zadanie 8. (5* punktów) (termin oddania 21 grudnia g. 15:45 przez formularz `https://classroom.github.com/a/U_8h3BI6`) Pod adresem `https://iidle.herokuapp.com/` jest pewna gra, dotycząca naszych studiów. Gra ta, choć ciekawa, jest dość żmudna - wiele razy trzeba wysyłać ten sam formularz. Twoim zadaniem będzie automatyzacja grania w tę grę. *Uwaga: To zadanie zostało napisane z wykorzystaniem przeglądarki Chrome. Pod Firefoksem jest niemal identycznie, a w innych przeglądarkach to nie wiem.*

Zapoznaj się z zasadami gry i pograj chwilę. Odkryjesz, że najbardziej uciążliwe jest wysyłanie planu dnia. Wejdź na stronę wysyłania planu, ale tym razem przed wysłaniem planu wciśnij F12, a następnie wybierz zakładkę „Network"[4]. Wyślij plan, a następnie zaobserwuj, co się stało. Pojawiło się nowe żądanie. Kliknij to żądanie prawym przyciskiem myszki, a następnie wybierz „Copy as cURL".

[2] git posiada dużo tego typu przydatnych narzędzi i nie ma sensu ich wszystkich omawiać na tym przedmiocie, więc przyglądamy się jednemu przykładowemu narzędziu.

[3] To ma oznaczać moment, w którym popełniono jakiś błąd.

[4] Oczywiście w innych wersjach językowych będzie się to inaczej nazywać.



Wklej ten tekst do jakiegoś edytora tekstu – jest to polecenie pozwalające wysłać, z pomocą programu cURL, plan do serwera gry. Stwórz na tej podstawie skrypt (w bashu, pythonie albo dowolnym innym języku) do grania w grę, który sam w odpowiednich odstępach czasu będzie wysyłał plany.

To zadanie ma 2 wersje: w podstawowej, za 4 punkty, nie należy się martwić autoryzacją i można założyć, że skrypt działa tak długo, jak autoryzacja jest ważna (czyli po jakimś czasie serwer przestanie przyjmować żądania - wtedy trzeba będzie ręcznie skopiować nowy token autoryzacji). Aby dostać piąty punkt, należy dopisać zarządzanie logowaniem i odświeżaniem tokenów autoryzacji tak, by skrypt grał w grę zupełnie sam.