Projekty - Programowanie Funkcyjne 2025

Swobodnie możesz korzystać z pomocy sztucznej inteligencji lub z dostępnych w sieci kodach. Ważne jest to aby rozumieć wszystkie fragmenty kodu. Kody źródłowe powinny być przepuszczone przez program hlint (ale nie stosuj automatycznie wszystkich wskazówek: kod powinien być czytelny przede wszystkim dla Ciebie).

Każdy z projektów wymaga (w różnym stopniu) skorzystania z mechanizmów interakcji z otoczeniem. Te zagadnienia nie były jeszcze omówione na wykładzie. Ale do ich realizacji wystarczy podstawowa wiedza zawarta np. na stronie https://www.haskell.org/tutorial/io.html. Pomóc Ci tu może również AI - w większości przypadków współczesne chaty poprawnie odpowiadają na pytania na temat podstawowych mechanizmów IO.

Projekty realizować macie w grupach maksymalnie 3 osobowych. Wybrany temat zgłaszamy prowadzącemu ćwiczenia.

1. **Automaty skończone**.
   Celem projektu jest wymodelowanie niedeterministycznego automatu skończonego, opracowanie procedury przekształcenia go w automat deterministyczny i minimalizacja przestrzeni stanów otrzymanego automatu. Możesz założyć, że przestrzenią stanów jest skończony podzbiór liczb naturalnych. Nie musisz się zbytnio przejmować optymalizacją kodu, ale zbudowana funkcja powinna sobie radzić bez trudu z automatami o 15 stanach.
2. **Maszyna licznikowa**
   Celem projektu jest zaimplementowanie Maszyny Licznikowej (Counter Machine) o kilku licznikach. Maszyna ta powinna wczytywać ciąg poleceń z podanej listy, symulować obliczenia i zwracać stan końcowy oraz ślad obliczeń w postaci listy stanów maszyny. Do generowania śladu obliczeń można skorzystać z monady Writer. Zaimplementuj operacje dodawania, mnożenia i funkcję sprawdzającą parzystość.
3. **Proca grawitacyjna**
   Napisz projekt ilustrujący zjawisko procy grawitacyjnej lub podwójnego wahadła. Efektem końcowym powinno być (minimum) coś w stylu projektu znajdującego się na stronie https://cs.pwr.edu.pl/cichon/2014_15_a/Grafika/Proca.html
   Do oprogramowania grafiki możesz zastosować bibliotekę Gloss języka Haskell.
4. **Program typu Eliza**
   Zaimplementuj swoją wersję programu Eliza wyspecjalizowaną w wybranej dziedzinie (np. w książkach SF, w modzie, i.t.p.). Program w trakcie rozmowy powinien gromadzić jakieś sensowne informacje o użytkowniku, a po zakończeniu dialogu powinien zapisać zebrane informacje o użytkowniku w jakimś pliku. Możesz prowadzić dialog w języku angielskim.
5. **Dithering**
   Zaimplementuj algorytm ditheringu (np. dithering Floyda-Steinberga). Skorzystaj z prostego formatu graficznego (np. TGA, PPM, XPM). Program powinien umożliwiać kontrolę parametrów -

np. liczbę bitów na kolor, metrykę w przestrzeni kolorów (np. euklidesowa, taksówkowa). Do obsługi plików graficznych możesz skorzystać np. z pakietów JuicyPixels lub Frida

6. **Lista TO-DO**.
   Aplikacja listy zadań do wykonania:Napisz aplikację listy zadań do wykonania w wierszu poleceń, w której użytkownicy mogą dodawać, usuwać i przeglądać zadania. Zadania powinny mieć termin wykonania. Lista ta powinna być zapisywana i wczytywana do pliku.
7. **Text adventure game**
   Stwórz grę przygodową opartą na tekście, w której gracze poruszają się po różnych pokojach i podejmują decyzje wpływające na wynik. Ten projekt pozwala Ci eksplorować rekurencyjne struktury danych i zarządzanie stanem.
8. **Web scraper z analizą danych** Opis: Pobieranie danych z internetu (np. kursy walut, pogoda) i ich analiza. Biblioteki: http-conduit, tagsoup, aeson (dla JSON). Rozszerzenia: Automatyczne raporty, wykresy.
9. **Prosty Web Scraper**
   Napisz program, który pobiera i analizuje kod HTML ze strony internetowej w celu wyodrębnienia określonych informacji, takich jak nagłówki lub linki.
10. **Analizator danych CSV**
    Opracuj narzędzie do wczytywania, filtrowania i analizowania danych z plików CSV. Powinno zawierać funkcje: średnie, mediany, wykresy (np. z Chart lub Diagrams). Super by było aby oprogramować interaktywną konsolę lub GUI.
11. **Narzędzie do badania grafów**
    Opracuj program, który dla danego grafu skończonego wyznaczy jego składowe, średnice jego składowych maksymalny i minimalny rząd wierzchołka, statystyki rzędów wierzchołków, podstawowe współczynniki klasteryzacji, statystyki odległości między wierzchołkami grafów. Graf powinien być wczytywany z pliku. Należy wybrać jakiś sensowny sposób zapisywania grafów i ich wewnętrznych reprezentacji. Powinien sobie radzić z grafami do 10^5 wierzchołków.
12. **Symulator gier planszowych**
    Zaimplementuj logikę gry takiej jak szachy, warcaby, Go lub prostszej (np. Tic-Tac-Toe). Super by było aby zaimplementowana była jakaś AI przeciwnika (minimax, alfa-beta pruning) i aby projekt był wyposażony w interfejs tekstowy lub graficzny.
13. **Kodowanie Huffmana**
    Napisz program wykorzystujący kodowanie Huffmana (klasyczne lub dynamiczne) do kompresji plików. Ten projekt pozwoli na potrenowanie operowania na danych binarnych oraz wykorzystania drzewiastych strukturach danych. Do obsługi danych binarnych możesz wykorzystać pakiet “bytestring”.
14. **Monada probabilistyczna**
    Zapoznaj się monadą probabilistyczną i dowiedz się jak się można ją posługiwać. Oprogramuj za pomocą tej monady paradoks Monty’ego Halla. Oto drugi problem: rozważamy ustalone schorzenie; prawdopodobieństwo zwracania poprawnej odpowiedzi przez dany medyczny test wynosi p. Prawdopodobieństwo, że dana osoba choruje na diagnozowaną chorobę wynosi q.

Napisz kalkulator, który obliczy prawdopodobieństwo choroby w przypadku pozytywnego wyniku testu.