Kolokwium 1

Programowanie w jezyku Rust Wersja B

Wszystkie zadania wykonujemy w jednym pliku. Powinna sie w nim znaleźć też funkcja `main()`, która przetestuje działanie kodu z zadań.

Prosze o podpisanie tej kartki jako listy obecności.

Zadanie 1 (0-5) Numer PESEL składa sie z 11 cyfr. Ostatnia jest cyfra kontrolna, umożliwiajaca kontrole poprawności. Napisz funkcje `fn pesel(&str) -> bool`, która zwróci *true*, jeżeli identyfikator wprowadzony w argumencie jest prawidłowym numerem PESEL. Sprawdź czy napis podany w parametrze:

- ma odpowiednia długość,
- składa sie wyłacznie z cyfr,
- ostatnia cyfra stanowi poprawna cyfre kontrolna.

Algorytm obliczania cyfry kontrolnej:

1. dla kolejnych dziesieciu cyfr obliczany jest iloczyn cyfry i jej indeksu (zaczynajac od 1),
2. obliczana jest suma tych iloczynów, oznaczmy ja S
3. obliczana jest reszta z dzielenia przez 10 liczby S, oznaczmy ja M
4. jeśli M = 10, to cyfra kontrolna równa jest 0
5. jeśli M != 10, to cyfra kontrolna wynosi S - M

Poprawne numery PESEL: 55030101193, 55030101230, 44051401458.

Zadanie 2 (0-5) Stwórz typ `ShoppingList`, reprezentujacy liste zakupów. W tym celu utwórz typ `Position`, składajacy sie z napisu (nazwy produktu) oraz wartości logicznej, mówiacej o tym, czy dany przedmiot został już zakupiony. Własnorecznie zaimplementuj dla niego ceche `PartialEq`, która bedzie ignorować wartość logiczna.

Typ `ShoppingList` powinien przechowywać wiele obiektów `Position`. Zaimplementuj dla niego metody:

- `fn new() -> ShoppingList`, tworzaca pusta liste,
- `fn push(Position)`, dodajaca element do listy,
- `fn bought(&str)`, oznaczajaca produkt o danej nazwie jako kupiony.

Ponadto, `ShoppingList` powinien posiadać metode `print(self)`, drukujaca wszystkie elementy listy, poczynajac od tych, które nie zostały jeszcze kupione.

Zadanie 3 (0-5) Stwórz typ `Book`, zawierajacy tytuł, autora, rok wydania oraz, opcjonalnie, tłumacza ksiażki.

Zaprojektuj typ `Library`, który bedzie przechowywał wiele ksiażek oraz informacje o tym czy dana pozycja jest dostepna do wypożyczenia. Zaimplementuj dla niego metody:

- `fn push_book(&mut self, b: Book)`, dodajaca nowa ksiażke do biblioteki,
- `fn find(&self, title: &str) -> Vec<Book>`, zwracajaca wektor wszystkich dostepnych ksiażek o danym tytule,
- `fn borrow(&mut self, b: Book) -> Result<Book, String>`, wypożyczajaca ksiażke, o ile jest ona dostepna,
- `fn return_book(&mut self, b: Book)`, oznaczajaca ksiażke jako dostepna.