Zadania domowe na pracownię nr 4

W tym tygodniu aż dwa zadania!

Zadanie A (Szyfr Cezara)

Szyfr Cezara polega na szyfrowaniu każdego znaku poprzez cykliczne przesunięcie jego wartości o k miejsc w alfabecie (w naszym wypadku **w prawo**). Liczbę k nazywamy *kluczem*. Przykładowo, zakładając, że alfabet składa się z łacińskich małych liter od a do z, kodowanie ciągu znaków `"aczx"` kluczem 1 daje ciąg `"bday"`.

Celem tego zadania jest zaimplementowanie szyfrowania i deszyfrowania dla dowolnego alfabetu, podanego jako lista wartości, które można porównać na równość racketowym predykatem `eq?` (np. liczby, symbole, `char`-y). Należy zaimplementować procedurę:

- `(define (caesar alphabet key) ... )`, gdzie:
  - `alphabet` to lista dowolnych wartości (ale takich, na których działa predykat `eq?`), będąca naszym alfabetem,
  - `key` to klucz, czyli liczba pozycji w alfabecie o które trzeba przesunąć w prawo dany znak, by go zaszyfrować, i w lewo, by go deszyfrować. Klucz może być dowolną liczbą całkowitą,
  - wynikiem powinna być **para procedur** `(cons encode decode)`, gdzie:
    - `(encode x)` szyfruje pojedynczy znak `x`,
    - `(decode x)` deszyfruje pojedynczy znak `x`.

Przykładowo:

```
> (define a-z (string->list "abcdefghijklmnopqrstuvwxyz"))
> (define ring (caesar a-z 1))
> (list->string (map (car ring) (string->list "abcxyz")))
"bcdyza"
> (list->string (map (cdr ring) (string->list "bcdyza")))
"abcxyz"
> (map (car (caesar (list 0 1 2 'a 'b 'c) 2))
       (list 0 2 'c 'c 1 2 'b 'a))
'(2 b 1 1 a b 0 c)
```

Wskazówka: Być może przydadzą się procedury `rotate` i `zip` zdefiniowane na ćwiczeniach.

Zadanie B (Łamanie Szyfru Cezara)

Być może 2000 lat temu szyfr Cezara był stosunkowo bezpieczny, ale tak naprawdę jest go dość łatwo złamać. Najprostszy atak wynika z obserwacji, że zaszyfrowany tekst zachowuje wiele statystycznych własności tekstu szyfrowanego. Przykładowo, najczęściej występujący znak tekstu, który szyfrujemy, kodowany jest przez znak, który jest najczęściej występującym znakiem tekstu zaszyfrowanego. W długiem tekście najczęściej występującym znakiem jest zwykle spacja, więc wystarczy znaleźć klucz, który dekoduje najczęściej występujący znak w tekście zaszyfrowanym na spację, i użyć go do dekodowania całego tekstu.

W tym zadaniu należy zdefinować procedurę, która robi właśnie to, ale dla dowolnego alfabetu i dowolnego kandydata na najczęściej występujący znak (jak pokazują przykłady poniżej, ten atak nie zawsze jest skuteczny). Zdefiniuj procedurę:

- `(define (crack-caesar alphabet c xs) ... )`, gdzie:
  - `alphabet` to alfabet (jak w poprzednim zadaniu),
  - `c` to kandydat na najczęściej występujący znak w tekście jawnym (np. spacja jeśli spodziewamy się, że deszyfrujemy standardowy tekst w języku polskim albo angielskim),
  - `xs` to lista reprezentująca tekst zaszyfrowany
  - wynikiem powinna być lista reprezentująca tekst będący kandydatem na deszyfrowany tekst jawny zgodnie z algorytmem opisanym powyżej.

```
> (define alphabet
    (let ([az (string->list "ąbcćdeęfghijklłmnńoópqrsśtuvwxyzźż")])
      (list* #\space #\. #\, #\! #\- #\: #\newline
             (append az (map char-upcase az)))))
> (define (crack-string xs)
    (list->string (crack-caesar alphabet #\space (string->list xs))))
> (define text
    "ŃukeĘsńwkeBvkĘGfeĘeqĄźłesńleśkteĘevźJtzfjKłGeĘGĘkumHGnexstmHowxleżsó
ńIeHsowsfjXketĆźĄośexsoewkeńzCnewsośBmkeńzeĘkutsjKxseńzCneńkĄxsfełGeBtĄGv
kewzqsvGj-GmrfemzeżzuoqxlęełlńIezńĆlńeHkJkĄĆlfjZeĘzuzewzśkfekułzeĘHqkĄńGe
ĘkĄĆlg")
```

```
> (display (crack-string text))
Dla widma sławy, w grób idą jak w łóżko,
Aby wywalczyć nikczemną piędź ziemi,
Na której nie ma dość miejsca do walki
Ani dość darni, by skryła mogiły
Tych, co polegną. Bądź odtąd zażartą,
O wolo moja, albo wzgardy wartą!
> (display (crack-string (list->string
                          (map (car (caesar alphabet 123))
                               (string->list "ala ma kota")))))
 ę Vg Vejo
```

Uwaga: Proszę założyć, że w tekście zaszyfrowanym istnieje tylko jedna najczęściej występująca wartość!

Wskazówka: Jest kilka sposobów, żeby odkryć, która wartość występuje w tekście zaszyfrowanym najczęściej. Najprostszym sposobem jest przejście przez całą listę budując słownik, czyli strukturę zawierającą pary kluczy (znakow) i wartości (liczba wystąpień), a następnie (np. przy użyciu odpowiednio zdefiniowanej procedury `maximum` z ćwiczeń), znaleźć najczęściej występujący element w wynikowym słowniku. W szablonie rozwiązania znajdą Państwo przykładową implementację słownika przy użyciu list. Można ich użyć, ale należy pamiętać o zachowaniu abstrakcji danych!

Uwagi

Pliki – każdy o nazwie `solution.rkt` – zawierające rozwiązania każdego z zadań należy przesłać w systemie Web-CAT dostępnym na SKOS-owej stronie przedmiotu w *nieprzekraczalnym* terminie **29 marca 2021 r., godz. 05.30**. Proszę pamiętać o klauzuli `provide` zgodnie z szablonami rozwiązań dostępnym na SKOS-ie. Pamiętaj o zasadach współpracy opisanych w regulaminie.