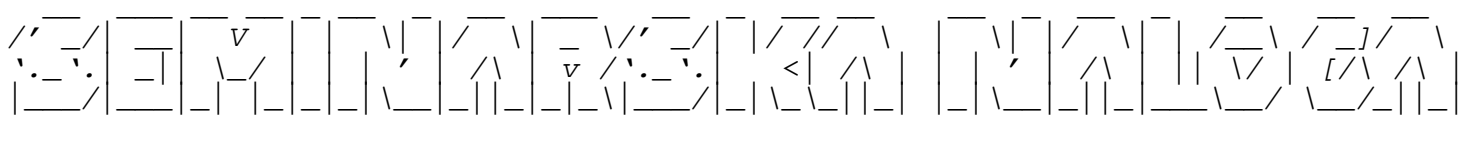
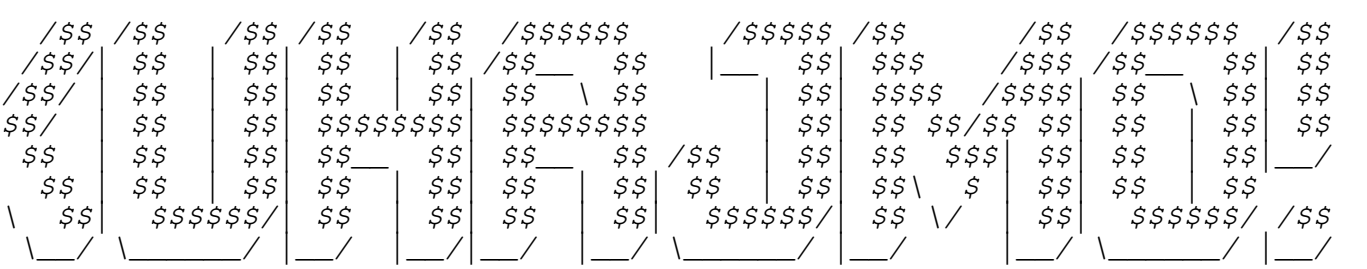
(*





Pri tej seminarski nalogi boste napisali program, ki vam bo pomagal pri delu s sestavinami, recepti, cenami in kuharskimi knjigami. Seminarska naloga je sestavljena iz dveh delov. V prvem delu boste implementirali podatkovno strukturo slovar (angl. dictionary), ki jo boste v drugem delu uporabili za implementacijo gurmanskih metod.

Seminar je sestavljen iz treh datotek:

- *cookbook-si.sml oz. cookbook-en.sml* — *V teh datotekah so zapisani podpisi, ki jih morate implementirati. Teh datotek NE SPREMINJAJTE.*

- *solution.sml* — *V tej datoteki implementirajte zahtevane strukture. Osnovna oblika datoteke je ze podana.*

- *tests.sml* — *V tej datoteki zapisite vase teste, ki naj cim bolje preizkusijo pravilnost delovanja vasih implementacij.*

Prilozena je se datoteka Makefile, ki je ze nastavljena tako, da pozene vase teste.

Tockovanje:

- *Dictionary: 20 tock*
- *Cookbook: 70 tock*
- *Zagovor: 10 tock*

Na spletni ucilnici oddajte datoteko z imenom <vpisna st.>-sem1.zip, v kateri naj bosta neposredno (brez poddirektorijev) shranjeni datoteki solution.sml in tests.sml.

```
*)




(* Podpis DICTIONARY definira podatkovni tip in programski vmesnik za delo
   s podatkovno strukturo slovar. Podatkovna struktura nam omogoca hranjenje
   podatkov v obliki parov (kljuc, vrednost) ter pridobivanje vrednosti prek
   kljuca. *)
signature DICTIONARY =
sig
    (* Podatkovni tip za delo s slovarjem. Definira tip kljuca, ki mora
       podpirati operacijo preverjanja enakosti, ter tip vrednosti, ki
       so shranjene v slovarju. *)
    type (''key, 'value) dict

    (* Vrne prazen slovar. *)
    val empty: (''key, 'value) dict

    (* Vrne true, ce v slovarju obstaja podatek s podanim kljucem. *)
    val exists: (''key, 'value) dict → ''key → bool

    (* Vrne true, ce je slovar prazen. *)
    val isEmpty: (''key, 'value) dict → bool

    (* Vrne stevilo podatkovnih parov (kljuc, vrednost) v slovarju. *)
    val size: (''key, 'value) dict → int

    (* Vrne SOME podatek, ki ustreza podanemu kljucu, ali NONE,
       ce kljuc v slovarju ne obstaja. *)
    val get: (''key, 'value) dict → ''key → 'value option

    (* Vrne podatek, ki ustreza podanemu kljucu, ali privzeto vrednost,
       ce kljuc v slovarju ne obstaja. *)
    val getOrDefault: (''key, 'value) dict → ''key → 'value → 'value

    (* Shrani par (kljuc, vrednost) v slovar in vrne posodobljen slovar.
       Ce par s podanim kljucem ze obstaja, prepise obstojeco vrednost. *)
    val set: (''key, 'value) dict → ''key → 'value → (''key, 'value) dict

    (* Odstrani podatek s podanim kljucem iz slovarja in vrne
       posodobljen slovar. *)
    val remove: (''key, 'value) dict → ''key → (''key, 'value) dict

    (* Vrne seznam kljucev v slovarju. *)
    val keys: (''key, 'value) dict → ''key list

    (* Vrne seznam vrednosti v slovarju. *)
    val values: (''key, 'value) dict → 'value list

    (* Vrne seznam parov (kljuc, vrednost) v slovarju. *)
    val toList: (''key, 'value) dict → (''key * 'value) list

    (* Izdela nov slovar iz seznama parov (kljuc, vrednost).
       Ce obstaja vec parov z istim kljucem, potem obvelja
       najbolj desna vrednost. *)
    val fromList: (''key * 'value) list → (''key, 'value) dict

    (* Zdruzi dva slovarja, tako da doda vrednosti iz desnega v levega
       in po potrebi prepise obstojece vrednosti v levem slovarju. *)
    val merge: (''key, 'value) dict
```

```
        → (''key, 'value) dict
        → (''key, 'value) dict

    (* Ohrani le tiste pare (kljuc, vrednost), za katere
       podana funkcija vrne true. *)
    val filter: ((''key * 'value) → bool)
        → (''key, 'value) dict
        → (''key, 'value) dict

    (* Preslika pare (kljuc, vrednost) s podano funkcijo. Ce funkcija
       ni injektivna glede na kljuc, je rezultat nedefiniran. *)
    val map: ((''key * 'value) → (''newkey * 'newvalue))
        → (''key, 'value) dict
        → (''newkey, 'newvalue) dict
end




(* Podpis COOKBOOK definira podatkovne tipe in programski vmesnik za delo
   s sestavinami, zalogo, recepti, cenami in kuharskimi knjigami. *)
signature COOKBOOK =
sig
    (* Podatkovni tip za sestavino. *)
    type ingredient

    (* Podatkovni tip za zalogo. Zaloga je preslikava iz sestavine v
       kolicino, ki je na zalogi. *)
    type stock

    (* Podatkovni tip za cenik. Cenik je preslikava iz sestavine v
       njeno ceno. *)
    type pricelist

    (* Podatkovni tip za recept. Recept opisuje sestavino, ki jo kuhamo,
       ter zalogo sestavin, ki jih za to potrebujemo. *)
    type recipe

    (* Podatkovni tip za knjigo receptov. *)
    type cookbook

    exception NoPriceException

    (* Ustvari novo sestavino iz podanega imena. *)
    val makeIngredient: string → ingredient

    (* Ustvari zalogo iz seznama parov (sestavina, kolicina). *)
    val makeStock: (ingredient * int) list → stock

    (* Ustvari cenik iz seznama parov (sestavina, cena). *)
    val makePricelist: (ingredient * real) list → pricelist

    (* Ustvari recept iz sestavine, ki je produkt recepta,
       ter zaloge potrebscin. *)
    val makeRecipe: (ingredient * stock) → recipe

    (* Ustvari knjigo receptov iz seznama receptov. *)
    val makeCookbook: recipe list → cookbook

    (* Pretvori sestavino v besedilo. *)
    val ingredientToString: ingredient → string
```

```
(* Pretvori zalogo sestavin v besedilo sledece oblike:

       imePrveSestavine: kolicinaPrveSestavine
       imeDrugeSestavine: kolicinaDrugeSestavine
       ...: ...

   Sestavine morajo biti urejene leksikografsko (vrstni red ASCII).
   Navedite le sestavine s pozitivno zalogo.
   Ne pozabite na znake za novo vrstico. *)
val stockToString: stock → string

(* Pretvori cenik v besedilo sledece oblike:

       imePrveSestavine: cenaPrveSestavine
       imeDrugeSestavine: cenaDrugeSestavine
       ...: ...

   Sestavine morajo biti urejene leksikografsko (vrstni red ASCII).
   Ne pozabite na znake za novo vrstico. *)
val pricelistToString: pricelist → string

(* Pretvori recept v besedilo sledece oblike:

       === imeRecepta ===
       imePrveSestavine: kolicinaPrveSestavine
       imeDrugeSestavine: kolicinaDrugeSestavine
       ...: ...

   Sestavine morajo biti urejene leksikografsko (vrstni red ASCII).
   Ne pozabite na znake za novo vrstico. *)
val recipeToString: recipe → string

(* Pretvori knjigo receptov v besedilo sledece oblike:

       === imePrvegaRecepta ===
       imePrveSestavine: kolicinaPrveSestavine
       imeDrugeSestavine: kolicinaDrugeSestavine
       ...: ...

       === imeDrugegaRecepta ===
       imePrveSestavine: kolicinaPrveSestavine
       imeDrugeSestavine: kolicinaDrugeSestavine
       ...: ...

   Recepti morajo biti urejeni leksikografsko (vrstni red ASCII).
   Sestavine v vsakem receptu morajo biti urejene
   leksikografsko (vrstni red ASCII).
   Ne pozabite na znake za novo vrstico. *)
val cookbookToString: cookbook → string

(* Vrne true, ce je v podani zalogi dovolj sestavin za podani recept. *)
val hasEnoughIngredients: stock → recipe → bool

(* Skuha recept in vrne posodobljeno zalogo z dodanim produktom recepta
   in odstranjenimi sestavinami tega recepta. *)
val cook: recipe → stock → stock

(* Vrne ceno zaloge - vsoto cen sestavin na zalogi glede na podani cenik.
   Ce katere koli sestavine ni na ceniku, naj funkcija vrne
   izjemo NoPriceException. *)
val priceOfStock: stock → pricelist → real

(* Vrne ceno recepta - vsoto cen potrebscin gnede na podani cenik.
```

```
    Ce katere koli potrebscine ni na ceniku, naj funkcija vrne
    izjemo NoPriceException. *)
  val priceOfRecipe: recipe → pricelist → real

  (* Za podani recept vrne zalogo potrebscin, ki jih nimamo
    na podani zalogi. Ce imamo na zalogi dovolj potrebscin,
    naj funkcija vrne prazno zalogo. *)
  val missingIngredients: recipe → stock → stock

  (* Vrne seznam receptov, ki jih lahko skuhamo s podano zalogo - recepte
    brez manjkajocih potrebscin. Recepti naj se obravnavajo neodvisno
    od drugih in ne zaporedno. Z drugimi besedami, predhodno obravnavani
    recepti nimajo vpliva na trenutno obravnavani recept. *)
  val possibleRecipes: cookbook → stock → cookbook

  (* Vrne kuharsko knjigo vseh receptov, ki jih lahko skuhamo s podanimi
    zamenjavami. Zamenjave so podane v ekvivalencnih razredih:

    [
        [piscanec, govedina, teletina, svinjina],
        [rdeca paprika, zelena paprika],
        ...
    ]

    Ce npr. recept vsebuje piscanca, ga lahko zamenjamo s svinjino.
    Ce npr. recept vsebuje zeleno papriko, jo lahko zamenjamo z rdeco.
    Zamenjav je lahko vec (paprika in meso).

    - Vrnjena kuharska knjiga naj vsebuje tudi recept brez zamenjav.
    - Recept lahko vsebuje sestavine, ki niso v nobenem ekvivalencnem
      razredu.
    - Predpostavljate lahko, da se v ekvivalencnem razredu sestavina
      pojavi le enkrat.
    - Predpostavljate lahko, da recept ne bo vseboval dveh sestavin iz
      enega ekvivalencnega razreda. *)
  val generateVariants: recipe → ingredient list list → cookbook

  (* Iz kuharske knjige vrne najcenejsi recept glede na podan cenik
    ali NONE, ce je kuharska knjiga prazna. *)
  val cheapestRecipe: cookbook → pricelist → recipe option
end
```