```
who= cs310 2[abc] Adam_Minter
here= /home/aminter1/Project2/310.2
total 56
 4 drwxr-xr-x  4 aminter1 aminter1  4096 2012-03-06 10:34 .
 4 drwxr-xr-x 10 aminter1 aminter1  4096 2012-03-04 00:57 ..
 4 -rw-r--r--  1 aminter1 aminter1   591 2012-03-06 10:06 2a.lisp
 4 -rw-r--r--  1 aminter1 aminter1  2639 2012-03-06 10:34 2b.lisp
 4 -rw-r--r--  1 aminter1 aminter1   412 2012-03-06 01:03 2c.lisp
 4 drwxr-xr-x  2 aminter1 aminter1  4096 2012-03-04 02:47 CS310-Supplements
 4 drwxr-xr-x  8 aminter1 aminter1  4096 2012-03-06 10:35 .git
12 -rw-r--r--  1 aminter1 aminter1 10707 2012-03-06 10:25 grammars.lisp
 4 -rw-r--r--  1 aminter1 aminter1   295 2012-01-29 22:34 main.lisp
 4 -rw-r--r--  1 aminter1 aminter1   174 2012-03-05 20:05 README
 4 -rw-r--r--  1 aminter1 aminter1  2731 2012-01-29 22:34 scifi.txt
 4 -rw-r--r--  1 aminter1 aminter1  3035 2012-01-29 22:34 story.lisp


______________________________________
running ...

;testing  !RANDS
;testing  !TIME-IT
;testing  !RANDOM-ELT
;testing  !GENERATE1
;testing  !GENERATE2
;testing  !GENERATE3

(ENGLISHMAJOR ENGL221 ENGL226 ENGL263 ENGL301 ENGL309 ENGL319 ENGL337 CS101
 ENGL241 ENGL242 ENGL261 SOCA221 ECON201 PHIL260 PHIL260 ENGL200 UNIV199
 ENGL101 ENGL102 ENGL132 PET101 SPAN204 SPAN203 SPAN102 SPAN101)
(MECHANICALANDAREOSPACEENGINEERINGMAJOR MAE316 MAE320 MAE335 MAE343 EE221 EE222
 MAE336 MAE345 MAE365 RELG219 MAE215 MAE241 MAE242 MAE243 MAE244 STAT215
 MATH251 ENGL102 PHYS112 ENGR102 MATH156 PHIL140 SOCA105 ENGR199 ENGR101
 ENGL101 CHEM115 MATH155)
(CHEMISTRYMAJOR MATH251 CHEM310 CHEM313 CHEM346 CHEM347 CHEM348 CHEM349 RELG219
 MATH156 CHEM233 CHEM234 CHEM235 CHEM236 ENGL102 MATH155 MATH153 MATH154 PET101
 COMM202 ECON201 PHYS111 PHYS122 UNIV199 ENGL101 CHEM115 CHEM116 CHEM117
 CHEM118 DANCE101 SOCA105 COM101)
(MECHANICALANDAREOSPACEENGINEERINGMAJOR MAE316 MAE320 MAE335 MAE343 EE221 EE222
 MAE336 MAE345 MAE365 HIST201 MAE215 MAE241 MAE242 MAE243 MAE244 STAT215
 MATH251 ENGL102 PHYS112 ENGR102 MATH156 FILM102 THET101 ENGR199 ENGR101
 ENGL101 CHEM115 MATH155)
(COMPUTERSCIENCEMAJOR CS310 CS350 CPE310 CPE311 CS450 CS450 CS101 SOCA221
 PHIL260 SOCA221 CS210 CS220 CS221 CS230 CPE271 CPE272 STAT215 MATH251 ENGL102
 PHYS112 CS110 CS111 MATH156 COM101 DANCE101 ENGR199 ENGR101 ENGL101 CHEM115
 MATH155)
(CHEMISTRYMAJOR MATH251 CHEM310 CHEM313 CHEM346 CHEM347 CHEM348 CHEM349 RELG219
 MATH156 CHEM233 CHEM234 CHEM235 CHEM236 ENGL102 MATH155 MATH153 MATH154
 PHIL140 RELG219 PHIL260 PHYS101 PHYS102 UNIV199 ENGL101 CHEM115 CHEM116
 CHEM117 CHEM118 THET101 PHIL140 COM101)
(MECHANICALANDAREOSPACEENGINEERINGMAJOR MAE316 MAE320 MAE335 MAE343 EE221 EE222
 MAE336 MAE345 MAE365 HIST201 MAE215 MAE241 MAE242 MAE243 MAE244 STAT215
 MATH251 ENGL102 PHYS112 ENGR102 MATH156 PET101 DANCE101 ENGR199 ENGR101
 ENGL101 CHEM115 MATH155)
(BIOLOGYMAJOR BIO321 BIO301 BIO338 BIO351 BIO361 BIO362 BIO363)
(INTERNATIONALSTUDIESMAJOR SPAN330 SPAN331 SPAN332 SPAN431 SPAN461 SPAN464
 ECON201 ECON202 SPAN203 SPAN204 SPAN301 SPAN302 GEO215 GEO243 PHIL260 FLIT113
 FLIT114 FLIT115 FLIT116 UNIV199 ENGL101 ENGL102 ENGL132 THET101 SPAN101
 SPAN102)
(HISTORYMAJOR HIST330 HIST331 HIST332 HIST334 HIST358 HIST359 HIST271 HIST272
 HIST220 HIST210 HIST221 ECON202 CS101 HIST101 HIST102 HIST104 HIST105 UNIV199
 ENGL101 ENGL102 PSYC101 PET101 SPAN204 SPAN203 SPAN102 SPAN101 RELG219 COMM202)

;testing  !RAINDI
;testing  !RESET-SEED
;testing  !SHUFFLE
;testing  !ONE-OF
;testing  !MAPPEND
;testing  !TERMINALP
;testing  !UNDEFINED-NONTERMINAL
;testing  !UNUSED-REWRITE
;testing  !GENERATES
```



```
(EARTH ISATTACKEDBY GIANT LUNAR BLOBS WANTSOMETHING)
(EARTH FALLSINTOSUN EVERYBODYDIE)
(EARTH ISATTACKEDBY TINY EXTRAGALACTIC SUPERBEINGS WANTSOMETHING)
(EARTH SCIENTISTS DISCOVER GIANT REPTILES WANTSOMETHING)
(EARTH ISSTRUCKBYAGIANT CLOUD ANDNOTDESTROYED EVERYBODYDIE)
(EARTH SCIENTISTS INVENT GIANT BUGS WANTSOMETHING)
(EARTH ISSTRUCKBYAGIANT COMET ANDISDESTROYED)
(EARTH FREEZES EVERYBODYDIE)
(EARTH BURNSUP EVERYBODYDIE)
(EARTH ISATTACKEDBY ENORMOUS MARTIAN BLOBS WANTSOMETHING)
;testing  !STORYCACHE

(AJIT WALKS VERY VERY QUICKLY SLOWLY WITH PIMA)
(BARKHA WALKS VERY VERY VERY VERY QUICKLY SLOWLY WITH PIMA)
(AJIT WALKS VERY QUICKLY SLOWLY WITH BARKHA)
(AJIT WALKS QUICKLY SLOWLY WITH AJIT)
(AJIT RUNS QUICKLY SLOWLY WITH AJIT)
(BARKHA WALKS QUICKLY SLOWLY WITH PIMA)
(BARKHA RUNS QUICKLY SLOWLY WITH PIMA)
(BARKHA WALKS VERY VERY VERY VERY QUICKLY SLOWLY WITH AJIT)
(AJIT RUNS VERY QUICKLY SLOWLY WITH AJIT)
(AJIT RUNS VERY VERY VERY QUICKLY SLOWLY WITH PIMA)
; pass : 15 = 100.0%
; fail : 0 =   0.0%
NIL



=====|  2a.lisp |============================
;***********Tests***************;
(deftest !generate3 ()
  (reset-seed)
  (test nil (generates 10 'schedule '*grammar3*)))

(deftest !raindi ()
  (reset-seed)
  (test 9 (randi 10)))

(deftest !reset-seed ()
  (test 10013 (reset-seed)))

(deftest !shuffle ()
  (reset-seed)
  (test '(adam bro dude brah) (shuffle '(bro dude adam brah))))

(deftest !random-elt ()
  (reset-seed)
  (test 'brah (random-elt '(bro dude adam brah))))

(deftest !one-of ()
  (reset-seed)
  (test '(brah) (one-of '(bro dude adam brah))))

(deftest !mappend ()
  (test '(2 3 5 6) (mappend #'cdr '((1 2 3) (4 5 6)))))

=====|  2b.lisp |============================
(defparameter *terms_Hash* (make-hash-table))

(defparameter *grammar* nil)

(defparameter *testgrammarFAIL*
  '((!Test -> (!Awesome !Dolphin !Basketball))
    (!Awesome -> ((adamMinterIsAwesome) !PaintingStuff))
    (!StupidTerm ->   )
    (!Lard ->  (iLiekTurtles) (iHeardULiekMudkips))
    (!Batman -> (betterHaveHisBackBrokenByBaneInDarkKnightRises))
    (!Dolphin -> (dolphinsRule))
    (!RickSantorum -> )))
```

```
(defun get-terms (gram)
  (setf *grammar* gram)
  (let ((terms nil) (nonterms nil) (rhs nil))
      (dolist (i *grammar*)
        (if (terminalp (car i))
            (push (car i) nonterms)
            nil)
        (let ((temp (flatten (rest (rest i)))))
          (dolist (j temp)
            (if (terminalp j)
                (push j rhs)
                (push j terms)))))
      (setf (gethash 'NonTerms *terms_Hash*) (remove nil
                                                     (reverse
                                                      (flatten
                                                       (remove-duplicates nonter
ms)))))
      (setf (gethash 'RHS_NonTerms *terms_Hash*) (remove-duplicates rhs))
      (setf (gethash 'Terminals *terms_Hash*) (remove nil (remove-duplicates ter
ms)))))



;1.) DefTests for Terminalp function;
(deftest !terminalp ()
  (test t (terminalp '!TEST)))

;I need to clarify something: When I initially wrote this function, and the scif
i grammar, I;
;wrote all of my NON-TERMINALS to have ! marks as the first character. I later l
earned that ;
;this isn't what I was supposed to do, but instead put an ! in front of all term
inals. The  ;
;code and my tests all still work the same, but I just test for non-terminals, i
nstead of   ;
;terminals. Sorry for any confusion...
            ;
(defun terminalp (x)
  (and (symbolp x)
       (eql (char (symbol-name x) 0) #\!)))



;2.) Deftests for Undefined-nonterminal function;
(deftest !undefined-nonterminal ()
  (test '(!HappyEnding !AndTakeAFewAndLeave) (undefined-nonterminal *grammar4*))
)

(defun undefined-nonterminal (gram)
  (setf *grammar* gram)
  (get-terms *grammar*)
  (let ((lst nil) (lhs (gethash 'NonTerms *terms_Hash*)) (rhs (gethash 'RHS_NonT
erms *terms_Hash*)))
    (dolist (i rhs)
      (if (null (member i lhs))
          (setf lst (list i lst))
          t))
    (remove nil (reverse (flatten lst)))))



;3.) Deftests for Unused-Rewrite Function;
(deftest !unused-rewrite ()
  (test '(!WantSomething !Dine !Denoument?) (unused-rewrites *grammar4*)))

(defun unused-rewrites (gram)
  (setf *grammar* gram)
  (get-terms *grammar*)
  (let ((lst nil) (lhs (gethash 'NonTerms *terms_Hash*)) (rhs (gethash 'RHS_NonT
erms *terms_Hash*)))
    (pop lhs)
    (dolist (i lhs)
      (if (null (member i rhs))
```



```
          (setf lst (list i lst))
          t))
    (remove nil (reverse (flatten lst)))))


=====|  2c.lisp |============================
(defparameter *grammar* nil)

(deftest !generates ()
  (reset-seed)
  (test nil (generates 10 'Start '*grammar4*)))

(deftest !storyCache ()
  (reset-seed)
  (storyCache 'BOY *grammar2*)
  (test nil (generates 10 'sentence '*grammar*)))

(defun storyCache (nonterm g)
  (setf *grammar* g)
  (let ((a (random-elt (cdr (cdr (assoc nonterm *grammar*))))))
  (rplacd (assoc nonterm *grammar*) (list '-> a))))



=====|  CS310-Supplements |============================


=====|  grammars.lisp |============================
(defparameter *grammar1*
  '((sentence -> (noun-phrase verb-phrase))
    (noun-phrase -> (Article Noun))
    (verb-phrase -> (Verb noun-phrase))
    (Article -> the a)
    (Noun -> man ball woman table)
    (Verb -> hit took saw liked))
  "A grammar for a trivial subset of English.")

(defparameter *grammar1*
  '((sentence -> (noun-phrase verb-phrase))
    (noun-phrase -> (Article Adj* Noun PP*) (Name) (Pronoun))
    (verb-phrase -> (Verb noun-phrase PP*))
    (PP* -> () (PP PP*))
    (Adj* -> () (Adj Adj*))
    (PP -> (Prep noun-phrase))
    (Prep -> to in by with on)
    (Adj -> big little blue green adiabatic)
    (Article -> the a)
    (Name -> He Her)
    (He   -> obama george)
    (Her  -> michelle laura)
    (Noun -> man ball woman table)
    (Verb -> hit took saw liked)
    (Pronoun -> He Her it these those that)))

(defparameter *grammar2*
        '((Sentence -> (Nounphrase Verbphrase))
          (Nounphrase -> Boy Girl)
          (Boy -> john ajit)
          (Girl -> pima barkha)
          (Verbphrase -> (Verb Modlist Adverb with  Nounphrase))
          (Verb -> runs  walks )
          (Modlist ->  () (very Modlist))
          (Adverb -> (quickly  slowly))))

(defparameter *grammar3*
        '((Schedule -> Major)
          (Major -> Science Arts)
          (Science -> Computer_Science Chemistry Mechanical_and_Aerospace_Enginee
ring Biology)
          (Arts -> English Philosphy History International_Studies)
```

```
            ;Standard set of GECs that all students must take.   ;
            ;                                                    ;
            ;                                                    ;
285         ;Also, it won't work in it's current state. I need   ;
            ;to basically implement the hash table functionality ;
            ;used in 2c to cache what GECs are already in the    ;
            ;course plan. If not, it will sometimes show a course;
            ;plan with a GEC listed twice. I'll keep it for now, ;
290         ;but when I get 2c finished, I'll try and fix 2a to  ;
            ;account for this. I'm thinking something along the  ;
            ;lines of what an attributed grammar does. ^_^       ;
            (GEC_1 -> soca105 phil140 dance101 geo110 com101 film102 pet101 thet101
     engl132 psyc101)
            (GEC_2 -> GEC_1 econ201 econ202 comm202 soca221 cs101 hist201 relg219 p
    hil260)
295


            ;*************Science based Course Plans**************;
            ;I broke up each type of major into specific clusters ;
            ;to make things easier to read. These clusters  are   ;
300         ;the standard classes that each engineering or medical;
            ;science student must take to graduate. I referenced  ;
            ;the WVU Course Catalog for this.


305         ;*****Engineering Major Cluster (CS and MAE)*******;
            (Engineering_Cluster0 -> (math156 GEC_1 GEC_1 EC0PreReqs0))
            (EC0PreReqs0 -> (engr199 engr101 engl101 chem115 math155))

            (Engineering_Cluster1 -> (stat215 (EC1PreReqs1)))
310         (EC1PreReqs1 -> (math251 engl102 phys112))

            ;Computer Science Course Plan;
            (Computer_Science -> (ComputerScienceMajor CSYear3))
            (CSYear3 -> (cs310 cs350 cpe310 cpe311 CS400xx CS400xx GEC_SubGroup CSY
    ear2))
315         (CSYear2 -> (cs210 cs220 cs221 cs230 cpe271 cpe272 Engineering_Cluster1
     CSYear1))
            (CSYear1 -> (cs110 cs111 Engineering_Cluster0))
            (CS400xx -> cs410 cs426 cs430 cs440 cs450 cs453 cs472 cs493)
            (GEC_SubGroup -> (GEC_2 GEC_2 GEC_2 GEC_2))

320         ;Mechanical & Aerospace Engineering Course Plan;
            (Mechanical_and_Aerospace_Engineering -> (MechanicalandAreospaceEnginee
    ringMajor MAEYear3))
            (MAEYear3 -> (mae316 mae320 mae335 mae343 ee221 ee222 mae336 mae345 mae
    365 GEC_2 MAEYear2))
            (MAEYear2 -> (mae215 mae241 mae242 mae243 mae244 Engineering_Cluster1 M
    AEYear1))
            (MAEYear1 -> (engr102 Engineering_Cluster0))
325
            ;*******Medical Science Clusters (Chemistry and Biology)*******;
            (Medical_Science_Cluster0 -> (phys111 phys122 MedSciPreReqs0) (phys101
    phys102 MedSciPreReqs0))
            (MedSciPreReqs0 -> (univ199 engl101 ((chem115 chem116) (chem117 chem118
    )) GEC_1))

330         (Medical_Science_Cluster1 -> (chem233 chem234 chem235 chem236 MedSciPre
    Reqs1))
            (MedSciPreReqs1 -> (engl102 (math155 (math153 math154)) GEC_1))

            ;Chemistry Course Plan;
            (Chemistry -> (ChemistryMajor ChemYear3))
335         (ChemYear3 -> (math251 chem310 chem313 chem346 chem347 chem348 chem349
    GEC_2 ChemYear2))
            (ChemYear2 -> (math156 Medical_Science_Cluster1 GEC_2 GEC_2 ChemYear1))
            (ChemYear1 -> (Medical_Science_Cluster0 GEC_1 GEC_1))

            ;Biology Course Plan;
340         (Biology -> (BiologyMajor BIOYear3))
            (BIOYear3 -> (bio321 BioFocuses))
            (BIOYear2 -> (stat211 bio219 bio221 Medical_Science_Cluster1 BIOYear1))
```



```
            (BIOYear1 -> (bio115 bio117 Medical_Science_Cluster0))

345         ;Biology has 4 unique sub focuses that students can take. The grouping;
            ;represents these focuses with their respective class.
    ;
            (BioFocuses -> Focus1 Focus2 Focus3 Focus4)
            (Focus1 -> (bio310 bio311 bio312 bio313 bio315 bio316 bio324 bio325))
            (Focus2 -> (bio336 bio337 bio339 bio340 bio341 bio348 bio350 bio352 bio
    353))
350         (Focus3 -> (bio301 bio338 bio351 bio361 bio362 bio363))
            (Focus4 -> (bio302 phys225 SubFocus1))
            (SubFocus1 -> agbi420 bioc339 bioc531)



355
            ;*************Art Major based Course Plans************ ;
            ;I started with the Science based course plans, as I am:
            ;more familiar with them. For an Art major, I've just  ;
            ;determined that each has their major requirements, and;
360         ;made up the other requirements. For example, I simply ;
            ;have all BA students take a language path, on top of  ;
            ;normal GECs.                                          :


365         ;The language GEC block for all Arts Student. ;
            ;I'm thinking I made throw this in some of the;
            ;Science major course plans.                 ;
            (Language_GEC -> Spanish Japanese)
            (Spanish -> (span204 PreReqSP204))
370         (PreReqSP204 -> (span203 PreReqSP203))
            (PreReqSP203 -> (span102 PreReqSP102))
            (PreReqSP102 -> span101)
            (Japanese -> (japn204 PreReqJP204))
            (PreReqJP204 -> (japn203 PreReqJP203))
375         (PreReqJP203 -> (japn102 PreReqJP102))
            (PreReqJP102 -> japn101)

            ;Basic required courses for all Arts students.  ;
            ;Trying to cut down on redundancy ^_^           ;
380         (Arts_Cluster0 -> (univ199 engl101 engl102 GEC_1 GEC_1))

            ;         English Course Plan         ;
            (English -> (EnglishMajor ENGLYear3))
            (ENGLYear3 -> (engl221 engl226 engl263 engl301 engl309 engl319 engl337
    GEC_2 ENGLYear2))
385         (ENGLYear2 -> (engl241 engl242 engl261 GEC_2 GEC_2 GEC_2 GEC_2 ENGLYear
    1))
            (ENGLYear1 -> (engl200 Arts_Cluster0 Language_GEC))

            ;         Philosphy Course Plan        ;
            (Philosophy -> (PhilosophyMajor PHILYear3))
390         (PHILYear3 -> (phil301 phil302 phil321 phil346 phil494 phil496 GEC_2 PH
    ILYear2))
            (PHILYear2 -> (phil244 phil248 phil260 GEC_2 GEC_2 PHILYear1))
            (PHILYear1 -> (Arts_Cluster0 Language_GEC GEC_1 GEC_2 GEC_2))

            ;         History Course Plan          ;
395         (History -> (HistoryMajor HISTYear3))
            (HISTYear3 -> (hist330 hist331 hist332 hist334 hist358 hist359 HISTYear
    2))
            (HISTYear2 -> (hist271 hist272 hist220 hist210 hist221 GEC_2 GEC_2 HIST
    Year1))
            (HISTYear1 -> (hist101 hist102 hist104 hist105 Arts_Cluster0 Language_G
    EC GEC_2 GEC_2))

400         ;   International Studies Course Plan  ;
            ;Contrary to most Arts majors, this    ;
            ;major is pretty well documented. They ;
            ;layout all kinds of different areas of;
            ;emphasis, but it's too much. There is ;
405         ;no way I'm coding them all just for   ;
            ;this project. I hope it doesn't cost  ;
            ;me any points lol. ^_^                ;
```

```
            ;                                       ;
            ;Also, something I need to point out is;
            ;that I specifically didn't add the    ;
            ;Language_GEC cluster to this, because ;
            ;I just chose "The Americas Required"  ;
            ;courses emphasis. Wouldn't make much  ;
            ;to randomly assign a language GEC with;
            ;"required" courses in Spanish if the  ;
            ;grammer randomly chooses Japanese -_- ;
            ;                                      ;
            ;That would totally be funny though.   ;
            (International_Studies -> (InternationalStudiesMajor ISYear3))
            (ISYear3 -> (span330 span331 span332 span431 span461 span464 ISYear2))
            (ISYear2 -> (econ201 econ202 span203 span204 span301 span302 geo215 geo
    243 GEC_2 ISYear1))
            (ISYear1 -> (flit113 flit114 flit115 flit116 Arts_Cluster0 span101 span
    102))))


    (defparameter *grammar4*
      '((Start -> (Earth !IsStressed))
        (!IsStressed -> !Catestrophes !Collision !Science !Attack)

        (!Catestrophes -> (!Catestrophe !PossibleMegaDeath))
        (!Collision -> (isStruckByAGiant !Floater !AndThen))
        (!Attack -> (isAttackedBy !Sizes !Extraterestrial !Beings !Whichetc))
        (!Science -> (scientists !DoScience !Sizes !Beings !Whichetc))

        ;Catestrophes;
        (!Catestrophe -> burnsUp freezes fallsIntoSun)

        ;Collision;
        (!Floater -> comet asteroid cloud)
        (!AndThen -> butIsSaved andIsDestroyed (andNotDestroyed !PossibleMegaDeath))
        (!PossibleMegaDeath -> everybodyDie (!SomeSaved !GoOn))
        (!SomeSaved -> somePeople everybody almostEverybody)
        (!GoOn -> dies !Rescued !Saved)

        ;Possible Collision Aftermath;
        (!Rescued -> (isRescued !Sizes !Extraterestrial !Beings))
        (!Saved -> butIsSavedBy !SomeOne !Science)

        (!DoScience -> invent discover)
        (!SomeOne -> earth !Extraterestrial)

        ;Attacks;
        (!Extraterestrial -> martian lunar extraGalactic)
        (!Beings -> bugs reptiles blobs superbeings)
        (!Sizes -> tiny giant enormous)
        (!Whichetc -> who WantSomething)
        (!WhichEtc -> )

        (!WantSomething -> !WantWomen (areFriendly !DenoumentOrHappyEnding) (!Unders
    tand !ButEtc))
        (!WantWomen -> (wantOurWomen !AndTakeAfewAndLeave))
        (!ButEtc -> (!AndAre radioactive !TryToKill))

        (!Understand -> (areFriendly butMisunderstood) misunderstandUs understandUsA
    llTooWell !Hungry)
        (!Hungry -> lookUponUsAsASourceOfNourishment)

        (!Dine -> (!Hungry and eat us Denoument?))

        (!AndAre -> andAre andAreNot)

        (!TryToKill -> (can be killed by !Killers) (can not be killed by !Killers !S
    oEtc))

        (!Killers -> !Killer (!Killer and !Killer))
        (!Killer -> aCrowdofPeasants theArmy theNavy theAirForce theMarines theCoast
    Guard theAtomBomb)

        (!SoEtc -> butTheyDieFromCatchingACold soTheyKillus soTheyPutUsUnderABenignD
```



```
    ictatorShip
            soTheyEatUs (soScientistsInventAWeapon !Which))
        (!Which -> whichTurnsThemIntoDisgustingLumps whichKillsThem (whichFails !SoE
    tc))

        (!Denoument? ->   !Denoument)
        (!Denoument? -> )

        (!DenoumentOrHappyEnding -> !Denoument !HappyEnding)
        (!Denoument -> (!Result !Ending))
        (!Result -> aCuteLittleKidConvincesThemPeopleAreOk aPriestTalksToThemOfGod t
    heyFallInLoveWithThisBeautifulGirl)

        (!Ending -> !Tragic !Happy)
        (!Tragic -> andTheyDie andTheyLeave andTheyTurnIntoDisgustingLumps)
        (!Happy -> andTheyGetMarriedAndLiveHappilyForeverAfter)))


    =====|  main.lisp |============================
    (handler-bind ((style-warning #'muffle-warning))
      (mapc 'load '(
                    "../tricks.lisp"
                    "grammars.lisp"
                    "story.lisp"
                    "2a.lisp"
                    "2b.lisp"
                    "2c.lisp"
                    )))

    (defun ! () (load "main.lisp"))

    (defun main ()
      (tests))


    (defun hello (&optional (who "world"))
           (format nil "hello ~a~%" who))


    =====|  README |============================
    menzies.us/cs310/cs310_20.html#Project2
    http://www.cs.cmu.edu/Groups/AI/html/cltl/clm/node153.html

    "What mappend should do:

    (mappend #'cdr '((1 2 3) (4 5 6)))

    (2 3 5 6)"




    =====|  scifi.txt |============================
    Start      -> Earth IsStressed
    IsStressed -> Catestrophes
    IsStressed -> Science
    IsStressed -> Attack
    IsStressed -> Collision

    Catestrophes -> Catestrophe and PossibleMegaDeath

    Catestrophe -> burnsUp
    Catestrophe -> freezes
    Catestrophe -> fallsIntoSun

    Collision -> isStruckByAGiant Floater AndThen

    Floater -> comet
    Floater -> asteroid
    Floater -> cloud

    AndThen -> butIsSaved
```

```
AndThen -> andIsDestroyed
AndThen -> andMagicallySaved


PossibleMegaDeath -> everybodyDies
PossibleMegaDeath -> Some GoOn

SomeSaved ->  somePeople
SomeSaved ->  everybody
SomeSaved ->  almostEverybody

GoOn -> dies
GoOn -> Resuced
GoOn -> Saved

Rescued -> isRescuedBy Sizes Extraterestrial Beings
Saved   -> butIsSavedBy SomeOne scientists the  Science

SomeOne -> earth
SomeOne -> extraterestrial

Science -> scientists DoSomething Sizes Beings Whichetc

DoSomething -> invent
DoSomething -> discover

Attack -> isAttackedBy Sizes Extraterestrial Beings Whichetc

Sizes -> tiny
Sizes -> giant
Sizes -> enormous

Extraterestrial -> martian
Extraterestrial -> lunar
Extraterestrial -> extraGalactic

Beings -> bugs
Beings -> reptiles
Beings -> blobs
Beings -> superbeings

Whichetc -> who WantSomething

WantSomething -> WantWomen
WantSomething -> areFriendly  and DenoumentOrHappyEnding
WantSomething -> UnderStand ButEtc

Understand -> areFriendly butMisunderstood
Understand -> misunderstandUs
Understand -> understandUsAllTooWell
Understand -> hungry

DenoumentOrHappyEnding -> Denoument
DenoumentOrHappyEnding -> HappyEnding

Dine -> Hungry and eat us Denoument?

WhichEtc ->
Hungry -> lookUponUsAsASourceOfNourishment

WantWomen -> wantOurWomen, AndTakeAFewAndLeave

ButEtc -> AndAre radioactive and TryToKill

AndAre -> andAre
AndAre -> andAreNot

Killers -> Killer
Killers -> Killer and Killer

Killer -> aCrowdOfPeasants
Killer -> theArmy
Killer -> theNavy
```



```
Killer -> theAirForce
Killer -> theMarines
Killer -> theCoastGuard
Killer -> theAtomBomb

TryToKill -> can be killed by Killers
TryToKill -> can not be killed by Killers SoEtc

SoEtc -> butTheyDieFromCatchingACold
SoEtc -> soTheyKillUs
SoEtc -> soTheyPutUsUnderABenignDictatorShip
SoEtc -> soTheyEatUs
SoEtc -> soScientistsInventAWeapon Which
SeEtc -> but Denoument

Which -> whichTurnsThemIntoDisgustingLumps
Which -> whichKillsThem
Which -> whichFails SoEtc

Denomument? ->
Denomument? -> Denoument

Denoument ->  aCuteLittleKidConvincesThemPeopleAreOk Ending
Denoument -> aPriestTalksToThemOfGod Ending
Denoument -> theyFallInLoveWithThisBeautifulGirl EndSadOrHappy

EndSadOrHappy -> Ending
EndSadOrHappy -> HappyEnding

Ending -> andTheyDie
Ending -> andTheyLeave
Ending -> andTheyTurnIntoDisgustingLumps

HappyEnding -> andTheyGetMarriedAndLiveHappilyForeverAfter



=====|  story.lisp |============================
; from http://changingminds.org/disciplines/storytelling/plots/propp/propp.htm

; lint (used, set, loops)
; memoization
; compilation
; meta-interpreter for maths
; compile meta interpreter into lambda bodies


;;;; -*- Mode: Lisp; Syntax: Common-Lisp -*-
;;; Code from Paradigms of Artificial Intelligence Programming
;;; Copyright (c) 1991 Peter Norvig

;;; ==============================

(defparameter *grammar* nil)

(defun random-elt (choices)
  (elt choices (randi (length choices))))

(defun one-of (set)
  (list (random-elt set)))

(defun mappend (fn list)
  "Append the results of calling fn on each element of list.
  Like mapcon, but uses append instead of nconc."
  (apply #'append (mapcar fn list)))

(defun combine-all (xlist ylist)
  "Return a list of lists formed by appending a y to an x.
  E.g., (combine-all '((a) (b)) '((1) (2)))
  -> ((A 1) (B 1) (A 2) (B 2))."
  (mappend #'(lambda (y)
               (mapcar #'(lambda (x) (append x y)) xlist))
           ylist))
```

```
(defun rule-lhs (rule)
  (first rule))

(defun rule-rhs (rule)
  (rest (rest rule)))

(defun rewrites (category)
  (rule-rhs (assoc category *grammar*)))

(defun generates (n phrase &optional (g *grammar*))
  (dotimes (i n)
    (setf *grammar* (eval g))
    (print (generate phrase))))

(defun generate (phrase)
  (cond ((listp phrase)
         (mappend #'generate phrase))
        ((rewrites phrase)
         (generate (random-elt (rewrites phrase))))
        (t (list phrase))))

(defun generate-tree (phrase)
  "return generate, and the rule name"
  (cond ((listp phrase)
         (mapcar #'generate-tree phrase))
        ((rewrites phrase)
         (cons phrase
               (generate-tree (random-elt (rewrites phrase)))))
        (t (list phrase))))

(defun generate-all (phrase)
  "exhaustive enumeratoion of all sentences.
   Warning: only use for short grammars"
  (cond ((null phrase) (list nil))
        ((listp phrase)
         (combine-all (generate-all (first phrase))
                      (generate-all (rest phrase))))
        ((rewrites phrase)
         (mappend #'generate-all (rewrites phrase)))
        (t (list (list phrase)))))

;;;;; tests
(deftest !random-elt ()
  (reset-seed)
  (test 'cabbage (random-elt '(apples bananas cabbage))))

(defun !generate1-prim  (root g &optional (n 5))
  (setf *grammar* g)
  (reset-seed)
  (mapcar #'generate
          '(sentence sentence sentence
            sentence sentence sentence sentence)))

(deftest !generate1 ()
  (test
   '((GEORGE SAW THESE)
     (GEORGE SAW A ADIABATIC GREEN WOMAN ON THE
         ADIABATIC BIG WOMAN IN A LITTLE BIG
             LITTLE BALL)
     (THE GREEN GREEN WOMAN IN LAURA HIT GEORGE WITH LAURA TO OBAMA)
     (THE WOMAN TOOK OBAMA IN LAURA) (GEORGE SAW THAT)
     (THE LITTLE GREEN TABLE WITH THESE WITH LAURA ON
          GEORGE IN GEORGE LIKED IT BY
          A MAN ON GEORGE BY LAURA TO A TABLE)
     (THE MAN SAW A MAN WITH THESE ON THE GREEN MAN IN THE ADIABATIC MAN))
    (!generate1-prim 'sentence *grammar1*)))

(deftest !generate2 ()
  (!generate1-prim 'sentence *grammar2*))
```