**ΑΝΑΦΟΡΑ PROJECT ΑΡΧΩΝ ΓΛΩΣΣΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΚΑΙ ΜΕΤΑΦΡΑΣΤΩΝ**

*Μέλη:*

Ζαροκανέλλος Γεώργιος 1072495 3ο έτος up1072495@upnet.gr

Σάμαρης Γεώργιος 1072493 3ο έτος up1072493@upnet.gr

Τριανταφυλλόπουλος Παναγιώτης 1072570 3ο έτος up1072570@upnet.gr

**Εξήγηση γραμματικής BNF**

**Program**: τα 2 αρχεία που δοκιμάσαμε τον μεταγλωττιστή αποτελούνται από μέρη που περιέχουν μέσα πολλά στοιχεία. Συγκεκριμένα το last\_result αποτελείται από τα last και active που είναι χωρισμένα με κόμμα συνεπώς το πρώτο μέρος του κανόνα θα είναι «T\_L\_BRACE({), [last](#last)(κανόνας) T\_COMMA [active](#active)(κανόνας) T\_R\_BRACE(})». Αντιθέτως το range\_result αποτελείται μόνο από το order(χρησιμοποιείται για να ελεγχθεί η σειρά των στοιχείων θα αναλυθεί παρακάτω) συνεπώς το δεύτερο μέρος του κανόνα είναι «T\_L\_BRACE({) order(κανόνας) T\_R\_BRACE(})» και το τελευταίο μέρος του κανόνα είναι για error recovery.

**expr**: Ως expr ορίζουμε τα στοιχεία που μπορούν να βρεθούν μέσα στο last, active και content των δύο αρχείων που δοκιμάζουμε. Όλα ακολουθούν το format «T\_QUOTATIONS(“) T\_(το ανάλογο token) T\_QUOTATIONS(“) T\_ASSIGNMENT(:) και έπειτα το ανάλογο token σχετικά με το τι τύπου στοιχεία είναι».

**\*****last**: σε αυτόν τον κανόνα ορίζουμε το πρώτο μεγάλο μέρος του last\_result το οποίο στο αρχείο βρίσκεται στην μορφή **“last” :** {last\_scope} άρα έχουμε «Τ\_QUOTATIONS (“) T\_LAST Τ\_QUOTATIONS (“) T\_ASSIGNMENT(:) T\_L\_BRACE({) last scope(κανόνας) T\_R\_BRACE(})». Ωστόσο στο range\_result το στοιχείο last είναι διαφορετικά ορισμένο για αυτό και ο κανόνας έχει δεύτερο μέρος. Μέχρι το T\_ASSIGNMENT είναι ίδιος ο κανόνας απλά αλλάζει το τελευταίο token που στην συγκεκριμένη περίπτωση είναι T\_BOOLEAN διότι το last στο αρχείο range\_result παίρνει boolean τιμή.

**last\_scope**: αφορά τον ορισμό των εσωτερικών στοιχείων του last για το αρχείο last\_result . Όπως περιγράψαμε παραπάνω τα στοιχεία που έχει το last τα έχουμε ορίσει ως expr. Τo last αποτελείται από expr που χωρίζονται με κόμμα συνεπώς ο κανόνας θα πρέπει να καλεί τον εαυτό του ξανά κάθε φορά που αναγνωρίζει κάποιο expr ακολουθούμενο από κόμμα και να τερματίζει αν διαβάσει μόνο expr όπου θα είναι και το τελευταίο στοιχείο του last. Συνεπώς ο κανόνας είναι «last\_scope : [expr](#expression) T\_COMMA last\_scope | expr |expr error last\_scope»(το τελευταίο μέρος του κανόνα είναι για error recovery σε περίπτωση που λείπει κάποιο κόμμα)

**\*****active**: σε αυτόν τον κανόνα ορίζεται το 2ο μεγάλο μέρος του last\_result το οποίο βρίσκεται στο αρχείο στην μορφή “active” : {active\_scope} άρα ο κανόνας θα είναι «Τ\_QUOTATIONS (“) T\_ACTIVE Τ\_QUOTATIONS (“) T\_ASSIGNMENT(:) T\_L\_BRACE({) active scope(κανόνας) T\_R\_BRACE(})».

**active\_scope**: Αφορά τον ορισμό των εσωτερικών στοιχείων του active. Όπως και το last έτσι και το active αποτελείται από expr χωρισμένες με κόμμα συνεπώς ο κανόνας θα είναι «active\_scope: [expr](#expression) T\_COMMA active\_scope | expr|expr error active\_scope;»(το τελευταίο μέρος του κανόνα είναι για error recovery σε περίπτωση που λείπει κάποιο κόμμα)

**\*u\_int\_array** : Οι πίνακες μη αρνητικών ακεραίων στα αρχεία που δοκιμάσαμε βρίσκονται στο αρχείο στην μορφή [u\_int\_scope] συνεπώς ο κανόνας θα είναι «T\_L\_BRACKET([) u\_int\_scope T\_R\_BRACKET(])»

**u\_int\_scope** : Οι πίνακες μη αρνητικών ακεραίων περιέχουν ακεραίους χωρισμένους με κόμμα συνεπώς ο κανόνας θα είναι «T\_U\_INT T\_COMMA(,) u\_int\_scope | T\_U\_INT | T\_U\_INT error u\_int\_scope»(ομοίως με τους παραπάνω κανόνες το τελευταίο μέρος του κανόνα είναι για error recovery στην έλλειψη κόμματος)

**json\_string**: τo json\_string βρίσκεται στο αρχείο στην μορφή “(όνομα του string)”. Επιπλέον θέλουμε το string active να το περάσουμε σαν τιμή στην μεταβλητή status συνεπώς ο κανόνας θα είναι «T\_QUOTATIONS(“) T\_JSON\_STRING T\_QUOTATIONS(“) | T\_QUOTATIONS(“) T\_ACTIVE T\_QUOTATIONS(“) | error».

**\*****json\_object**: το json αντικείμενο περιέχει πολλά στοιχεία μέσα του και βρίσκεται στο αρχείο στην μορφή {json\_obj\_scope} συνεπώς ο κανόνας θα είναι «T\_L\_BRACE({) json\_obj\_scope T\_R\_BRACE(}) | error»

**Json\_obj\_scope:** το json αντικείμενο περιέχει expressions χωρισμένα με κόμμα συνεπώς ο κανόνας θα είναι «[expr](#expression) T\_COMMA json\_obj\_scope | expr | expr error json\_obj\_scope» (ομοίως με τους παραπάνω κανόνες το τελευταίο μέρος του κανόνα είναι για error recovery στην έλλειψη κόμματος)

**\*json\_array:** το json array είναι της μορφής [json\_arr\_scope] συνεπώς ο κανόνας θα είναι «T\_L\_BRACKET([) json\_arr\_scope T\_R\_BRACKET(]) | error»

**Json\_arr\_scope**: το json array αποτελείται από json objects χωρισμένα με κόμμα συνεπώς ο κανόνας θα είναι : «[json\_object](#json_obj) T\_COMMA(,) json\_arr\_scope | json\_object | json\_object error json\_arr\_scope(ομοίως με πριν το τελευταίο μέρος του κανόνα είναι για error recovery)»

**\*****Content**: το content βρίσκεται στο αρχείο στην μορφή “content”: [content\_scope] άρα ο κανόνας θα είναι «T\_QUOTATIONS(“) T\_CONTENT T\_QUOTATIONS(“) T\_ASSIGNMENT(:) T\_L\_BRACKET([) content\_scope T\_R\_BRACKET(])»

**content\_scope:** το content περιέχει json objects χωρισμένα με κόμμα συνεπώς ο κανόνας θα είναι «[json\_object](#json_obj) T\_COMMA(,) content\_scope | json\_object | json\_object error content\_scope(ομοίως το τελευταίο μέρος είναι για error recovery)»

**wager\_objects:** το wager\_objects περιλαμβάνει τα στοιχεία που περιέχονται στο wager\_statistics δηλαδή τα columns, wagers και add\_on χωρισμένα με κόμμα και σε αυτήν την σειρά. Συνεπώς ο κανόνας θα είναι «columns T\_COMMA(,) wagers T\_COMMA(,) add\_on | error». Τα columns, wagers και add on τα ορίσαμε όλα σε διαφορετικούς κανόνες διότι μας ενδιαφέρει η σειρά.

**columns:** το columns είναι θετικός ακέραιος και βρίσκεται στο αρχείο στην μορφή “columns”:(u\_int) συνεπώς ο κανόνας θα είναι «T\_QUOTATIONS(“) T\_COLUMNS T\_QUOTATIONS(“) T\_ASSIGNMENT(:) T\_U\_INT | error»

**wagers:** το wagers είναι και αυτό θετικός ακέραιος και βρίσκεται στο αρχείο στην μορφή “wagers”:(u\_int) συνεπώς θα είναι «T\_QUOTATIONS(“) T\_WAGERS T\_QUOTATIONS(“) T\_ASSIGNMENT(:) T\_U\_INT | error»

**add\_on:** το add\_on είναι Json\_array και βρίσκεται στο αρχείο στην μορφή “addOn”: [] συνεπώς ο κανόνας θα είναι «T\_QUOTATIONS(“) T\_ADDON T\_QUOTATIONS(“) T\_ASSIGNMENT(:) T\_L\_BRACKET([) T\_R\_BRACKET(]) | error»

**price\_points:** περιγράφει το τι περιλαμβάνει το pricePoints το οποίο είναι το στοιχείο amount το οποίο είναι θετικός πραγματικός και βρίσκεται στο αρχείο στην μορφή {“amount”:(u\_float)} συνεπώς ο κανόνας θα είναι «T\_L\_BRACE({) Τ\_QUOTATIONS(“) T\_AMOUNT T\_QUOTATIONS(“) T\_ASSIGNMENT(:) T\_U\_FLOAT T\_R\_BRACE(}) | error»

**\*winning\_numbers:** το winning numbers περιέχει το εσωτερικά στοιχεία του winningNumbers(που έχουμε ορίσει σαν [expr](#expression)) τα οποία είναι τα list και bonus. Το εσωτερικό του winningNumbers είναι στην μορφή {list(κανόνας),bonus(κανόνας)} συνεπώς ο κανόνας θα είναι «T\_L\_BRACE({) list T\_COMMA(,) bonus T\_R\_BRACE(}) | error»

**list:** το List είναι το ένα από τα δύο στοιχεία που περιέχει το winningNumbers και είναι array που περιέχει θετικούς ακεραίους και βρίσκεται στο αρχείο στην μορφή “list”:[u\_int\_array] συνεπώς ο κανόνας θα είναι «T\_QUOTATIONS(“) T\_LIST T\_QUOTATIONS(“) T\_ASSIGNMENT(:) u\_int\_array | error»

**bonus:** το bonus είναι το δεύτερο στοιχείο που περιέχει το winningNumbers και είναι επίσης array που περιέχει ένα θετικό ακέραιο και βρίσκεται στο αρχείο στην μορφή “bonus” : [u\_int] συνεπώς ο κανόνας θα είναι «T\_QUOTATIONS(“) T\_BONUS T\_QUOTATIONS(“) T\_ASSIGNMENT(:) u\_int\_array | error»

Τα **total\_pages, total\_elements, number\_of\_elements, sort , first, size** και **number** ακολουθούν όλα το ίδιο format το οποίο είναι : “(totalPages|totalElements|…|size)” : (αντίστοιχη τιμή κάθε στοιχείου) συνεπώς οι κανόνες τους ακολουθούν την μορφή «T\_QUOTATIONS(“) αντίστοιχο token T\_QUOTATIONS(“) T\_ASSIGNMENT(:) αντίστοιχο token για τύπο τιμής» *(Τα στοιχεία αυτά παρόλο που έχουν το ίδιο format με τα* [*expr*](#expression) *επιλέξαμε να μην τα βάλουμε εκεί διότι χρειαζόμαστε να υπάρχουν με κάποια σειρά στο αρχείο)*

**order:** Στο αρχείο range\_result χρειαζόμαστε τα στοιχεία που περιέχει να εμφανίζονται σε μία συγκεκριμένη σειρά για αυτό και δημιουργήσαμε αυτόν τον επιπλέον κανόνα για να μπορούμε να την ελέγχουμε. Συγκεκριμένα πρέπει να εμφανίζονται τα στοιχεία με την εξής σειρά : content, totalPages, totalElements, last, numberOfElements, sort, first, size, number συνεπώς ο κανόνας θα είναι «[content](#content) T\_COMMA(,) total\_pages T\_COMMA(,) total\_elements T\_COMMA(,) last T\_COMMA(,) number\_of\_elements T\_COMMA(,) sort T\_COMMA(,) first T\_COMMA(,) size T\_COMMA(,) number | error»

**\*(Τα στοιχεία αυτά επιλέξαμε να χρησιμοποιήσουμε 2 κανόνες για να τα περιγράψουμε, έναν κανόνα για το ίδιο το στοιχείο και έναν για τα στοιχεία που περιέχει στο εσωτερικό του και αυτό για να εξαλείψουμε τα shift/reduce και reduce/reduce error)**

**Screenshots από εκτελέσεις**

**1ο screenshot (επιτυχία και error για last\_result)**

**Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα**

**2ο screenshot(επιτυχία και error για range\_result)**

**Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα**

**3ο screenshot(error από 3ο ερώτημα)**

**Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα**

**Κώδικας lexer(αρχείο L)**

%{

#include "parser.tab.h"

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <math.h>

#include <string.h>

extern int token\_numbers[2][10]; // counts tokens in last\_scope and active\_scope

extern void listIntegerInterval(int val); //defined in parser.y

extern int prizeCatElem[2][8]; //counts elements in prizeCategory

int lineno = 1; //line counter for error recovery

int value; //value of last integer read

int listflag = 0; // is 1 when reading list

int prizeCatflag = 0; // is 1 when reading prizeCategory

int contentflag = 0; // is 1 when reading content

%}

%option case-insensitive

%option noyywrap

digit [0-9]

letter [a-zA-Z]

sign [-,+]

u\_int {digit}+

float ({sign}{u\_int}+\.{u\_int}+)+

u\_float ({u\_int}+\.{u\_int}+)+

JSON\_string ({u\_int}|{letter}|\\_?|\.?)+

newline [\n]

%%

"{" {return T\_L\_BRACE;}

"}" {return T\_R\_BRACE;}

"[" {return T\_L\_BRACKET;}

"]" {return T\_R\_BRACKET;}

"." {return T\_DOT;}

\" {return T\_QUOTATIONS;}

":" {return T\_ASSIGNMENT;}

"," {return T\_COMMA;}

"last" {return T\_LAST;}

"gameId" {token\_numbers[1][0]++;return T\_GAMEID;}

"drawId" {token\_numbers[1][1]++;return T\_DRAWID;}

"drawTime" {token\_numbers[1][2]++;return T\_DRAWTIME;}

"status" {token\_numbers[1][3]++;return T\_STATUS;}

"drawBreak" {token\_numbers[1][4]++;return T\_DRAWBREAK;}

"visualDraw" {token\_numbers[1][5]++;return T\_VISUALDRAW;}

"pricePoints" {token\_numbers[1][6]++;return T\_PRICEPOINTS;}

"amount" {return T\_AMOUNT;}

"bonus" {return T\_BONUS;}

"prizeCategories" {token\_numbers[1][7]++;prizeCatflag=1;return T\_PRIZECATEGORIES;}

"id" {prizeCatElem[1][0]++; return T\_ID;}

"divident" {prizeCatElem[1][1]++; return T\_DIVIDENT;}

"winners" {prizeCatElem[1][2]++; return T\_WINNERS;}

"distributed" {prizeCatElem[1][3]++; return T\_DISTRIBUTED;}

"jackpot" {prizeCatElem[1][4]++; return T\_JACKPOT;}

"fixed" {prizeCatElem[1][5]++; return T\_FIXED;}

"categoryType" {prizeCatElem[1][6]++; return T\_CATEGORYTYPE;}

"gameType" {prizeCatElem[1][7]++; return T\_GAMETYPE;}

"minimumDistributed" {return T\_MINIMUMDISTRIBUTED;}

"wagerstatistics" {token\_numbers[1][8]++; return T\_WAGERSTATISTICS;}

"columns" {return T\_COLUMNS;}

"wagers" {return T\_WAGERS;}

"addOn" {return T\_ADDON;}

"active" {return T\_ACTIVE;}

"content" {contentflag=1;return T\_CONTENT;}

"winningNumbers" {token\_numbers[1][9]++;return T\_WINNINGNUMBERS;}

"list" {listflag=1;return T\_LIST;}

"totalPages" {return T\_TOTALPAGES;}

"totalELements" {return T\_TOTALELEMENTS;}

"numberOfElements" {return T\_NUMBEROFELEMEMENTS;}

"sort" {return T\_SORT;}

"direction" {return T\_DIRECTION;}

"true" {return T\_BOOLEAN;}

"false" {return T\_BOOLEAN;}

"property" {return T\_PROPERTY;}

"ignoreCase" {return T\_IGNORE\_CASE;}

"nullHandling" {return T\_NULL\_HANDLING;}

"descending" {return T\_DESCENDING;}

"ascending" {return T\_ASCENDING;}

"first" {return T\_FIRST;}

"size" {return T\_SIZE;}

"number" {return T\_NUMBER;}

{u\_int} {value = atoi(yytext);if(listflag==1)listIntegerInterval(value);return T\_U\_INT;} /\* set value equal to integer parsed and if reading "list" make sure it is in [1,45] \*/

{u\_float} {return T\_U\_FLOAT;}

{float} {return T\_FLOAT;}

{JSON\_string} {return T\_JSON\_STRING;}

{sign} {return T\_SIGN;}

{newline} {lineno++;}

%%

**Κώδικας parser(αρχείο Y)**

%{

#include <stdio.h>

#include <string.h>

#define EXIT\_FAILURE -1

void yyerror(const char \*s);

void prizeCatLimit(int counter);

void validateGameId(int val);

void listIntegerCountLimit(int counter);

void listIntegerInterval(int val);

void countAllElements();

void countActiveElements();

void winningNElements();

void bonusElements();

void counterChecker();

void prizeCatElements();

extern FILE \*yyin;

extern FILE \*yyout;

extern int yylex();

extern int lineno; //line counter

extern int value; //stores the value of last read integer

extern int listflag; //1 when reading a "list"

extern int prizeCatflag; //1 when reading a "prizeCategories"

extern int contentflag; //1 when reading a "content"

int token\_numbers[2][10]; //an array of tokens we expect to read in no particular order in last, active and content

int prizeCatElem[2][8]; //an array of tokens we expect to read in no particular order in prizeCategories

int minDistFlg = 0; //1 when reading "minimumDistributed"

int prCatCntr=0; //counts json objects in "prizeCategories"

int errors = 0; //counts errors encountered

int intArrCounter = 0; //counts integers in an array

%}

%token T\_U\_INT "Unsigned Integer"

%token T\_U\_FLOAT "Unsigned Float"

%token T\_GAMEID "GameID"

%token T\_FLOAT "Float"

%token T\_JSON\_STRING "JSON String"

%token T\_BOOLEAN "Boolean"

%token T\_JSON\_ARRAY "JSON Array"

%token T\_L\_BRACE "Left Brace ({)"

%token T\_R\_BRACE "Right Brace (})"

%token T\_L\_BRACKET "Left Bracket ([)"

%token T\_R\_BRACKET "Right Bracket (])"

%token T\_DOT "Dot (.)"

%token T\_QUOTATIONS "Quotations"

%token T\_ASSIGNMENT "Assignment (:)"

%token T\_COMMA "Comma (,)"

%token T\_U\_INT\_ARRAY "Unsigned Integer Array"

%token T\_LAST "Last"

%token T\_ACTIVE "Active"

%token T\_DRAWID "DrawID"

%token T\_DRAWTIME "Draw Time"

%token T\_VISUALDRAW "Visual Draw"

%token T\_PRICEPOINTS "Price Points"

%token T\_AMOUNT "Amount"

%token T\_WINNINGNUMBERS "Winning Numbers"

%token T\_LIST "List"

%token T\_BONUS "Bonus"

%token T\_PRIZECATEGORIES "Prize Categories"

%token T\_ID "ID"

%token T\_DIVIDENT "Divident"

%token T\_WINNERS "Winners"

%token T\_DISTRIBUTED "Distributed"

%token T\_JACKPOT "Jackpot"

%token T\_FIXED "Fixed"

%token T\_CATEGORYTYPE "Category Type"

%token T\_GAMETYPE "Game Type"

%token T\_MINIMUMDISTRIBUTED "Minimum Distributed"

%token T\_WAGERSTATISTICS "Wager Statistics"

%token T\_COLUMNS "Columns"

%token T\_WAGERS "Wagers"

%token T\_SORT "Sort"

%token T\_DIRECTION "Direction"

%token T\_NUMBEROFELEMEMENTS "Number of Elements"

%token T\_TOTALELEMENTS "Total Elements"

%token T\_TOTALPAGES "Total Pages"

%token T\_DRAWBREAK "Draw Break"

%token T\_STATUS "Status"

%token T\_CONTENT "Content"

%token T\_ADDON "Addon"

%token T\_PROPERTY "Property"

%token T\_IGNORE\_CASE "Ignore Case"

%token T\_NULL\_HANDLING "Null Handling"

%token T\_DESCENDING "Descending"

%token T\_ASCENDING "Ascending"

%token T\_FIRST "First"

%token T\_SIZE "Size"

%token T\_NUMBER "Number"

%token T\_SIGN "sign"

%token T\_EOF 0

%define parse.error verbose

%start program

%%

program: T\_L\_BRACE last T\_COMMA active T\_R\_BRACE

| T\_L\_BRACE order T\_R\_BRACE

| error

;

last\_scope: expr T\_COMMA last\_scope

| expr

| expr error last\_scope

;

last: T\_QUOTATIONS T\_LAST T\_QUOTATIONS T\_ASSIGNMENT T\_L\_BRACE last\_scope T\_R\_BRACE {countAllElements();}

| T\_QUOTATIONS T\_LAST T\_QUOTATIONS T\_ASSIGNMENT T\_BOOLEAN

| error

;

order: content T\_COMMA total\_pages T\_COMMA total\_elements T\_COMMA last T\_COMMA number\_of\_elements T\_COMMA sort T\_COMMA first T\_COMMA size T\_COMMA number

| error

;

active: T\_QUOTATIONS T\_ACTIVE T\_QUOTATIONS T\_ASSIGNMENT T\_L\_BRACE active\_scope T\_R\_BRACE {countActiveElements();}

| error

;

active\_scope: | expr T\_COMMA active\_scope

| expr

| expr error active\_scope

;

wager\_objects: columns T\_COMMA wagers T\_COMMA add\_on

| error

;

columns: T\_QUOTATIONS T\_COLUMNS T\_QUOTATIONS T\_ASSIGNMENT T\_U\_INT

| error

;

wagers: T\_QUOTATIONS T\_WAGERS T\_QUOTATIONS T\_ASSIGNMENT T\_U\_INT

| error

;

add\_on: | T\_QUOTATIONS T\_ADDON T\_QUOTATIONS T\_ASSIGNMENT T\_L\_BRACKET T\_R\_BRACKET

| error

;

price\_points: T\_L\_BRACE T\_QUOTATIONS T\_AMOUNT T\_QUOTATIONS T\_ASSIGNMENT T\_U\_FLOAT T\_R\_BRACE

| error

;

list: T\_QUOTATIONS T\_LIST T\_QUOTATIONS T\_ASSIGNMENT u\_int\_array {counterChecker(intArrCounter, 5);intArrCounter=0;listflag=0;}

| error

;

bonus: T\_QUOTATIONS T\_BONUS T\_QUOTATIONS T\_ASSIGNMENT u\_int\_array {counterChecker(intArrCounter, 1);intArrCounter=0;}

| error

;

winning\_numbers: T\_L\_BRACE list T\_COMMA bonus T\_R\_BRACE

| error

;

u\_int\_array: T\_L\_BRACKET u\_int\_scope T\_R\_BRACKET

| error

;

u\_int\_scope: T\_U\_INT T\_COMMA u\_int\_scope {intArrCounter++;}

| T\_U\_INT {intArrCounter++;}

| T\_U\_INT error u\_int\_scope

;

json\_string: T\_QUOTATIONS T\_JSON\_STRING T\_QUOTATIONS

| T\_QUOTATIONS T\_ACTIVE T\_QUOTATIONS

| error

;

json\_array: T\_L\_BRACKET json\_arr\_scope T\_R\_BRACKET

| error

;

json\_arr\_scope: json\_object T\_COMMA json\_arr\_scope

| json\_object

| json\_object error json\_arr\_scope

;

json\_object: T\_L\_BRACE json\_obj\_scope T\_R\_BRACE {if(prizeCatflag==0 && contentflag==1)countAllElements();if(prizeCatflag==1){prCatCntr++;prizeCatElements();}if(minDistFlg!=0){printf("Unexpected amount of \"minimumDistributed\" before line: %d\n", lineno);errors++;}minDistFlg=0;}

| error

;

json\_obj\_scope: expr T\_COMMA json\_obj\_scope

| expr

| expr error json\_obj\_scope

;

content\_scope: json\_object T\_COMMA content\_scope

| json\_object

| json\_object error content\_scope

;

content: T\_QUOTATIONS T\_CONTENT T\_QUOTATIONS T\_ASSIGNMENT T\_L\_BRACKET content\_scope T\_R\_BRACKET {contentflag=0;}

;

total\_pages: T\_QUOTATIONS T\_TOTALPAGES T\_QUOTATIONS T\_ASSIGNMENT T\_U\_INT

| error

;

total\_elements: T\_QUOTATIONS T\_TOTALELEMENTS T\_QUOTATIONS T\_ASSIGNMENT T\_U\_INT

| error

;

number\_of\_elements: T\_QUOTATIONS T\_NUMBEROFELEMEMENTS T\_QUOTATIONS T\_ASSIGNMENT T\_U\_INT

| error

;

sort: T\_QUOTATIONS T\_SORT T\_QUOTATIONS T\_ASSIGNMENT json\_array

| error

;

expr: T\_QUOTATIONS T\_GAMEID T\_QUOTATIONS T\_ASSIGNMENT T\_U\_INT {validateGameId(value);}

| T\_QUOTATIONS T\_DRAWID T\_QUOTATIONS T\_ASSIGNMENT T\_U\_INT

| T\_QUOTATIONS T\_DRAWTIME T\_QUOTATIONS T\_ASSIGNMENT T\_U\_INT

| T\_QUOTATIONS T\_STATUS T\_QUOTATIONS T\_ASSIGNMENT json\_string

| T\_QUOTATIONS T\_DRAWBREAK T\_QUOTATIONS T\_ASSIGNMENT T\_U\_INT

| T\_QUOTATIONS T\_VISUALDRAW T\_QUOTATIONS T\_ASSIGNMENT T\_U\_INT

| T\_QUOTATIONS T\_PRICEPOINTS T\_QUOTATIONS T\_ASSIGNMENT price\_points

| T\_QUOTATIONS T\_WINNINGNUMBERS T\_QUOTATIONS T\_ASSIGNMENT winning\_numbers

| T\_QUOTATIONS T\_PRIZECATEGORIES T\_QUOTATIONS T\_ASSIGNMENT json\_array {prizeCatLimit(prCatCntr);prCatCntr=0;prizeCatflag=0;}

| T\_QUOTATIONS T\_WAGERSTATISTICS T\_QUOTATIONS T\_ASSIGNMENT T\_L\_BRACE wager\_objects T\_R\_BRACE

| T\_QUOTATIONS T\_ID T\_QUOTATIONS T\_ASSIGNMENT T\_U\_INT {if(value < 1 || value > 9){errors++; printf("Expected a value from 1-8 got %d near line %d\n", value, lineno);} else if(value == 1) minDistFlg++;}

| T\_QUOTATIONS T\_DIVIDENT T\_QUOTATIONS T\_ASSIGNMENT T\_U\_FLOAT

| T\_QUOTATIONS T\_WINNERS T\_QUOTATIONS T\_ASSIGNMENT T\_U\_INT

| T\_QUOTATIONS T\_DISTRIBUTED T\_QUOTATIONS T\_ASSIGNMENT T\_U\_FLOAT

| T\_QUOTATIONS T\_JACKPOT T\_QUOTATIONS T\_ASSIGNMENT T\_U\_FLOAT

| T\_QUOTATIONS T\_FIXED T\_QUOTATIONS T\_ASSIGNMENT T\_U\_FLOAT

| T\_QUOTATIONS T\_CATEGORYTYPE T\_QUOTATIONS T\_ASSIGNMENT T\_U\_INT {if(value != 0 && value != 1){errors++; printf("Expected 0 or 1 got %d near line %d\n", value, lineno);}}

| T\_QUOTATIONS T\_GAMETYPE T\_QUOTATIONS T\_ASSIGNMENT json\_string

| T\_QUOTATIONS T\_MINIMUMDISTRIBUTED T\_QUOTATIONS T\_ASSIGNMENT T\_U\_FLOAT {minDistFlg--;}

| T\_QUOTATIONS T\_DIRECTION T\_QUOTATIONS T\_ASSIGNMENT json\_string

| T\_QUOTATIONS T\_PROPERTY T\_QUOTATIONS T\_ASSIGNMENT json\_string

| T\_QUOTATIONS T\_IGNORE\_CASE T\_QUOTATIONS T\_ASSIGNMENT T\_BOOLEAN

| T\_QUOTATIONS T\_NULL\_HANDLING T\_QUOTATIONS T\_ASSIGNMENT json\_string

| T\_QUOTATIONS T\_DESCENDING T\_QUOTATIONS T\_ASSIGNMENT T\_BOOLEAN

| T\_QUOTATIONS T\_ASCENDING T\_QUOTATIONS T\_ASSIGNMENT T\_BOOLEAN

| error

;

first: T\_QUOTATIONS T\_FIRST T\_QUOTATIONS T\_ASSIGNMENT T\_BOOLEAN

| error

;

size: T\_QUOTATIONS T\_SIZE T\_QUOTATIONS T\_ASSIGNMENT T\_U\_INT

| error

;

number: T\_QUOTATIONS T\_NUMBER T\_QUOTATIONS T\_ASSIGNMENT T\_U\_INT

| error

;

%%

// count # of tokens in last, active and content

int token\_numbers[2][10]={T\_GAMEID, T\_DRAWID, T\_DRAWTIME, T\_STATUS, T\_DRAWBREAK, T\_VISUALDRAW, T\_PRICEPOINTS, T\_PRIZECATEGORIES, T\_WAGERSTATISTICS, T\_WINNINGNUMBERS};

//count of elements in prizeCategories in range\_result

int prizeCatElem[2][8] = {T\_ID, T\_DIVIDENT, T\_WINNERS, T\_DISTRIBUTED, T\_JACKPOT, T\_FIXED, T\_CATEGORYTYPE, T\_GAMETYPE};

void yyerror(const char \*s) {

errors++;

printf("%s line: %d\n", s, lineno);

}

void validateGameId(int val){

//if value of gameId is not invalid

if(val!=1100 && val!=1110 && val!=2100 && val!=2101 && val!=5103 && val!=5104 && val!=5106)

{

printf("Invalid gameId: %d near line: %d\n", val, lineno);

errors++;

}

}

//validates that "prizeCategory" contains 8 json objects

void prizeCatLimit(int counter){

if(counter != 8 ){

errors++;

printf("Expected 8 JSON objects in prizeCategories found: %d, near line: %d\n",counter,lineno);

}

}

//counts elements in "prizeCategory"

void prizeCatElements()

{

for(int i = 0; i < 8; i++){

//each element must appear only once

if(prizeCatElem[1][i] != 1){

errors++;

printf("Found %d %s expected 1 before line: %d\n",prizeCatElem[1][i], yytname[YYTRANSLATE(prizeCatElem[0][i])], lineno);

}

//reset array

prizeCatElem[1][i]=0;

}

}

//integers in "list" must be in [1,45]

void listIntegerInterval(int val){

if(!(val >= 1 && val <= 45) ){

errors++;

printf("Expected integer in [1, 45] found: %d, near line: %d\n", val, lineno);

}

}

//counts elements in "last" for last\_result and in "content" for range\_result

void countAllElements(){

for(int i=0; i<10; i++){

//contains each token specified in token\_numbers[0][0:9] once

if(token\_numbers[1][i] != 1){

errors++;

printf("Found %d %s expected 1 before line: %d\n", token\_numbers[1][i], yytname[YYTRANSLATE(token\_numbers[0][i])], lineno);

}

//reset array

token\_numbers[1][i]=0;

}

}

//counts elements in "active" for file last\_result

void countActiveElements(){

for(int i=0; i<9; i++){

//contains each token specified in token\_numbers[0][0:8] once

if(token\_numbers[1][i] != 1){

errors++;

printf("Found %d %s expected 1 in \"active\" before line: %d\n", token\_numbers[1][i], yytname[YYTRANSLATE(token\_numbers[0][i])], lineno);

}

//reset array

token\_numbers[1][i] = 0;

}

//does not contain "winningNumbers"

if(token\_numbers[1][9] != 0){

errors++;

printf("Found %d %s expected 0 in \"active\" before line: %d\n", token\_numbers[1][9], yytname[YYTRANSLATE(token\_numbers[0][9])], lineno);

//reset array

token\_numbers[1][9] = 0;

}

}

//validates # of entries in an array

void counterChecker(int counter, int limit)

{

//counter: # integers read

//limit: # expected integers in an array

if(counter != limit){

printf("Found %d integers expected %d before line: %d\n", counter, limit, lineno);

errors++;

}

}

int main ( int argc, char \*\*argv )

{

++argv; --argc;

if ( argc > 0 )

yyin = fopen( argv[0], "r" );

else

yyin = stdin;

yyout = fopen ( "output", "w" );

yyparse ();

fclose(yyout);

fclose(yyin);

if(errors == 0)

printf("\n\nParsing successful \n\n");

else

printf("\nParsing failed due to %d errors \n\n", errors);

return 0;

}