


# Αρχές Γλωσσών Προγραμματισμού & Μεταφραστών 2021 - 2022


---

## Πίνακας περιεχομένων

Στοιχεία Φοιτητών.....	2
Γενικά.....	3
Ερώτημα 1.....	8
a) BNF γραμματική και συντακτικό για τη γλώσσα .....	8
b) Έλεγχος στοιχείων “last” & “active” .....	16
Ερώτημα 2 .....	17
Ερώτημα 3 .....	18
Υποερώτημα a .....	18
Υποερώτημα b.....	18
Υποερώτημα c .....	18
Παραδοχές – Σχόλια – Οδηγίες .....	19
Κώδικας .....	20
Flex.....	20
Bison .....	22

## Στοιχεία Φοιτητών

 Δασκαλάκης Ευάγγελος  
AM: 1079327      Έτος: 3<sup>ο</sup>      e-mail: [e\\_daskalakis@upnet.gr](mailto:e_daskalakis@upnet.gr)

 Μπουρνάκας – Δρακόπουλος Ίων  
AM:1075475      Έτος: 3<sup>ο</sup>      e-mail: [up1075475@upnet.gr](mailto:up1075475@upnet.gr)

 Χαλάτση Σταυρούλα  
AM: 1072619      Έτος: 3<sup>ο</sup>      e-mail: [chalatsi\\_s@upnet.gr](mailto:chalatsi_s@upnet.gr)

## Γενικά

### Λεκτική Ανάλυση (Lexical Analysis)

---

Στην αρχή του κώδικα του lexer αρχείου «myParser.l», με χρήση της εντολής `#include`, ενσωματώνονται στο πρόγραμμα τα απαραίτητα αρχεία επικεφαλίδων (header files). Σε αυτά τα αρχεία συμπεριλαμβάνεται και το αρχείο “y.tab.h”, το οποίο συμβάλλει στην αντιστοιχία των λεκτικών μονάδων με τα tokens.

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "y.tab.h"
%}
```

Έπειτα ορίζονται τα παρακάτω options: `noyywrap` και `yylineno`. Η option `noyywrap` αναγκάζει το lexer να σταματήσει όταν φτάσει στο πρώτο end-of-file (EOF). Με την option `yylineno`, δημιουργείται ένας σαρωτής και διατηρείται έτσι ο αριθμός της τρέχουσας γραμμής που διαβάζεται στην καθολική μεταβλητή `yylineno`.

```
%option noyywrap
%option yylineno
```

Στις παρακάτω γραμμές κώδικα περιέχονται διάφοροι ορισμοί αναγνωριστικών, δηλαδή γίνεται ανάθεση ονομάτων σε σύνολα από αναγνωριστικά, τα οποία είναι ορισμένα ως Κανονικές Εκφράσεις.

```
digit      [0-9]
property   (["][a-zA-Z .][a-zA-Z0-9 .]*["])
minID      [1]
IDdigit    [2-8]
boolean    [0-1]
boolean1    true|false
integer    "0" | {digit}+
quotation  ["]
identifier (["][a-zA-Z][a-zA-Z0-9]*["])
whitespace [ \t\n\f]
float      ({digit}+.{digit}+)
real       ([-]{float}|[-]{integer})
```

Στη συνέχεια, όπως φαίνεται στον παρακάτω κώδικα, ορίζονται τα απαραίτητα tokens και operators που αναγνωρίζει το myParser.l.

```
%%
\"last\"          {return T_LAST;}
\"active\"        {return T_ACTIVE;}
\"content\"       {return T_CONTENT;}

\"gameId\"        {return T_GAMEID;}
\"drawId\"        {return T_DRAWID;}
\"drawTime\"      {return T_DRAWTIME;}
\"status\"        {return T_STATUS;}
\"drawBreak\"     {return T_DRAWBREAK;}
\"visualDraw\"    {return T_VISUALDRAW;}
\"pricePoints\"   {return T_PRICEPOINTS;}
\"amount\"        {return T_AMOUNT;}
\"winningNumbers\" {return T_WINNINGNUMBERS;}
\"prizeCategories\" {return T_PRIZECATEGORIES;}
\"wagerStatistics\" {return T_WAGERSTATISTICS;}

\"list\"          {return T_LIST;}
\"bonus\"         {return T_BONUS;}

\"id\"            {return T_ID;}
\"divident\"      {return T_DIVIDENT;}
\"winners\"       {return T_WINNERS;}
\"distributed\"   {return T_DISTRIBUTED;}
\"jackpot\"       {return T_JACKPOT;}
\"fixed\"         {return T_FIXED;}
\"categoryType\" {return T_CATEGORYTYPE;}
\"gameType\"     {return T_GAMETYPE;}
\"minimumDistributed\" {return T_MINIMUMDISTRIBUTED;}

\"columns\"       {return T_COLUMNS;}
\"wagers\"        {return T_WAGERS;}
\"addOn\"         {return T_ADDON;}

\"totalPages\"    {return T_TOTALPAGES;}
\"totalElements\" {return T_TOTALELEMENTS;}

\"first\"         {return T_FIRST;}
\"numberOfElements\" {return T_NUMBEROFELEMENTS;}
\"sort\"         {return T_SORT;}
\"size\"         {return T_SIZE;}
\"number\"       {return T_NUMBER;}
\"direction\"    {return T_DIRECTION;}
```

```

"\property\"      {return T_PROPERTY;}
"\ignoreCase\"    {return T_IGNORECASE;}
"\nullHandling\"  {return T_NULLHANDLING;}
"\descending\"    {return T_DESCENDING;}
"\ascending\"     {return T_ASCENDING;}

"{"               {return T_LCURLY;}
"}"               {return T_RCURLY;}
","               {return T_COMMA;}
":"               {return T_COLON;}
"["               {return T_LBRACKET;}
"]"               {return T_RBRACKET;}

[ \t\n\v\b]      ;      /* skip whitespace */

{minID}           {return T_MINID;}
{IDDigit}         {return T_IDDIGIT;}
{boolean}         {return T_BOOLEAN;}
{boolean1}        {return T_BOOLEAN1;}
{integer}         {return T_INTEGER;}

{quotation}       {return T_Q;}
{identifier}      {
    yyval.str = strdup(yytext);
    return T_IDENTIFIER;
}

{property}        {
    yyval.str = strdup(yytext);
    return T_PROPERTY1;
}

{float}           {return T_FLOAT;}
{real}            {return T_REAL;}

{whitespace}      {/*ignore whitespace*/}
%%

```

## Συντακτική Ανάλυση (Syntax Analysis)

Στην αρχή του κώδικα του bison αρχείου «*myParser.y*», ενσωματώνονται στο πρόγραμμα τα απαραίτητα αρχεία επικεφαλίδων (header files). Επιπλέον, πραγματοποιούνται δηλώσεις μεταβλητών, οι οποίες θα χρειαστούν και στους δύο αναλυτές, με χρήση της extern. Σημαντικές είναι οι συναρτήσεις *yyparse()*, *yylex()*

και `yyerror(const char* s)`. Η συνάρτηση `yyparse()` είναι υπεύθυνη για την υλοποίηση του συντακτικού αναλυτή και επιστρέφει την τιμή «0», εάν αναγνωριστεί η συμβολοσειρά εισόδου, και την τιμή «1» εάν υπάρξει συντακτικό λάθος. Η συνάρτηση `yylex()` καλείται από την `yyparse()` κάθε φορά που πρέπει να διαβαστεί μια λεκτική μονάδα από τη συμβολοσειρά εισόδου και να αναγνωριστεί. Τέλος, η συνάρτηση `yyerror(const char* s)` καλείται αυτόματα όταν εντοπιστεί συντακτικό σφάλμα.

```
%{

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

extern int yylex();
extern int yyparse();
extern FILE *yyin;
extern FILE *yyout;
extern int yylineno;
extern int yywrap;

int yylex();
void yyerror(const char* s);
%}
```

Ακολουθώς δηλώνουμε τα tokens τα οποία ορίστηκαν ήδη από το λεκτικό αναλυτή.

```
%token T_Q
%token T_LAST
%token T_COLON
%token T_INTEGER
%token T_IDENTIFIER
%token T_FLOAT
%token T_GAMEID
%token T_DRAWID
%token T_DRAWTIME
%token T_STATUS
%token T_DRAWBREAK
%token T_VISUALDRAW
%token T_PRICEPOINTS
%token T_AMOUNT
%token T_WINNINGNUMBERS
%token T_PRIZECATEGORIES
%token T_WAGERSTATISTICS
%token T_LIST
```

```

%token T_BONUS
%token T_ID
%token T_DIVIDENT
%token T_WINNERS
%token T_DISTRIBUTED
%token T_JACKPOT
%token T_FIXED
%token T_CATEGORYTYPE
%token T_GAMETYPE
%token T_MINIMUMDISTRIBUTED
%token T_COLUMNS
%token T_WAGERS
%token T_ADDON
%token T_COMMA
%token T_LBRACKET
%token T_RBRACKET
%token T_LCURLY
%token T_RCURLY
%token T_ACTIVE
%token T_IDDIGIT
%token T_BOOLEAN
%token T_MINID
%token T_DIGIT
%token T_REAL

%union {
char* str;
}

```

Στο αμέσως επόμενο block του bison αρχείου, περιέχονται οι κανόνες του parser. Ενδεικτικά, παρακάτω φαίνονται μερικοί κανόνες. Ο πρώτος κανόνας start αποτελεί την αρχή του συντακτικού δέντρου και σε περίπτωση που η ανάλυση είναι επιτυχής, εκτυπώνεται συγκεκριμένο μήνυμα. Στον κανόνα start «καλούνται» οι αμέσως επόμενοι σε σειρά στο συντακτικό δέντρο κανόνες, οι οποίοι με τη σειρά τους «καλούν» επιπλέον κανόνες, επεκτείνοντας έτσι το συντακτικό δέντρο.

```

// JSON initialization
start:
    T_LCURLY last active T_RCURLY    {printf("\t~~~\t\t\nParsed
successfully!\n\t~~~");}
    | T_LCURLY content T_RCURLY    {printf("\t~~~\t\t\nParsed
successfully!\n\t~~~");}
    ;

```

Στο τελευταίο block βρίσκεται η συνάρτηση yyerror(const char\* s), η οποία εντοπίζει το πρώτο κατά σειρά συντακτικό λάθος και εμφανίζει το κατάλληλο μήνυμα, καθώς

και τη γραμμή όπου εντοπίστηκε το σφάλμα. Η main αναλαμβάνει να ανοίξει το προς ανάλυση αρχείο (τύπου JSON στις περιπτώσεις μας) με δικαιώματα read-only και να εκτελέσει parsing επί αυτού.

```
// Find the faulty line
void yyerror(const char* s)
{
    fprintf(stderr, "\t!!!\nParser error in line %d.\n\t!!!", yylineno);
    exit(1);
}

// Main function
int main (int argc, char **argv) {

    yyin = fopen(argv[1], "r"); // read permission
    yyparse ();

    return 0;
}
```

## Ερώτημα 1

a) BNF γραμματική και συντακτικό για τη γλώσσα

### Βασικοί όροι

(από αρχείο flex)

<digit>	::=	0   1   2   3   4   5   6   7   8   9
<letter>	::=	a   b   ... → ...   z   A   B   ... → ...   Z
<integer>	::=	<digit>   <digit> <integer>
<quotation>	::=	"
<string>	::=	<letter> <string>   <digit> <string>   <digit>   <letter>



$\langle \text{identifier} \rangle ::= \langle \text{quotation} \rangle ( \langle \text{letter} \rangle \mid \langle \text{string} \rangle )$   
 $\langle \text{quotation} \rangle$   
 $\langle \text{float} \rangle ::= \langle \text{integer} \rangle . \langle \text{integer} \rangle$   
 $\langle \text{real} \rangle ::= - \langle \text{float} \rangle \mid - \langle \text{integer} \rangle$   
 $\langle \text{symbols} \rangle ::= . \mid - \mid \{ \mid \} \mid [ \mid ] \mid , \mid :$

## Ειδικοί όροι

(προκύπτουν από τους Βασικούς όρους)

$\langle \text{boolean} \rangle ::= 0 \mid 1$   
 $\langle \text{boolean1} \rangle ::= \text{false} \mid \text{true}$   
 $\langle \text{IDdigit} \rangle ::= 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8$   
 $\langle \text{minID} \rangle ::= 1$   
 $\langle \text{propertyClass} \rangle ::= \langle \text{quotation} \rangle ( \langle \text{letter} \rangle . \mid \langle \text{string} \rangle . )$   
 $\langle \text{quotation} \rangle$

Οι ειδικοί όροι προέκυψαν κατ' ανάγκη από τα δεδομένα περιείχαν τα αρχεία για το συγκεκριμένο project. Ως όροι πάντως, καλύπτονται πλήρως από τους **Βασικούς**.

## Ειδικές συναρτήσεις

(στοιχεία των γενικότερων συναρτήσεων)

Καθώς όλα τα στοιχεία εντός των JSON αρχείων εμφανίζονταν με το όνομά τους, περάστηκαν και με τον ίδιο τρόπο στο συντακτικό αναλυτή, περικλείοντας το κάθε όνομα με τον τύπο:

`"\"some_random_name\""`

Στο BNF θα εμφανίζονται ως αναγνωρισμένα token με τον εξής τρόπο:

### Token

με το όνομα “token” να αντιστοιχεί κάθε φορά στο στοιχείο που περιγράφεται.

<gameId>	::=	<u>gameId</u> : <integer> ,
<drawId>	::=	<u>drawId</u> : <integer> ,
<drawTime>	::=	<u>drawTime</u> : <integer> ,
<status>	::=	<u>status</u> : <identifier> ,
<drawBreak>	::=	<u>drawBreak</u> : <integer> ,
<visualDraw>	::=	<u>visualDraw</u> : <integer> ,
<list>	::=	<u>list</u> : [ <arrayOfList> ] ,
<bonus>	::=	<u>bonus</u> : [ <integer> ]
<id>	::=	<u>id</u> : <IDdigit> ,
<minimumID>	::=	<u>id</u> : <minID> ,
<divident>	::=	<u>divident</u> : <float> ,
<winners>	::=	<u>winners</u> : <integer> ,
<distributed>	::=	<u>distributed</u> : <float> ,
<minimumDistributed>	::=	<u>minimumDistributed</u> : <float>
<jackpot>	::=	<u>jackpot</u> : <float> ,
<fixed>	::=	<u>fixed</u> : <float> ,
<categoryType>	::=	<u>categoryType</u> : <boolean> ,
<minGameType>	::=	<u>gameType</u> : <identifier> ,

<gameType>	::=	<u>gameType</u> : <identifier>
<columns>	::=	<u>columns</u> : <integer> ,
<wagers>	::=	<u>wagers</u> : <integer> ,
<addOn>	::=	<u>addOn</u> : [ <jsonElement> ]   : [ ]
<totalPages>	::=	<u>totalPages</u> : <integer> ,
<totalElements>	::=	<u>totalElements</u> : <integer> ,
<lastBoolean>	::=	<u>last</u> : <boolean1> ,
<numberOfElements>	::=	<u>numberOfElements</u> : <integer> ,
<first>	::=	<u>first</u> : <boolean1> ,
<size>	::=	<u>size</u> : <integer> ,
<number>	::=	<u>number</u> : <integer>
<direction>	::=	<u>direction</u> : <identifier> ,
<property>	::=	<u>property</u> : <propertyClass> ,
<ignoreCase>	::=	<u>ignoreCase</u> : <boolean1> ,
<nullHandling>	::=	<u>nullHandling</u> : <identifier> ,
<descending>	::=	<u>descending</u> : <boolean1> ,
<ascending>	::=	<u>ascending</u> : <boolean1> ,

## Γενικότερες συναρτήσεις

(περιέχουν τα πιο ειδικά Tokens)

`<pricePoints> ::= pricePoints : { amount : <float> },`

`<winningNumbers> ::= winningNumbers : { <list> <bonus> },`

`<prizeCategories> ::= prizeCategories : [ <minPrizeCategories>  
<defaultPrizeCategories> ,  
<defaultPrizeCategories> ,  
<defaultPrizeCategories> ,  
<defaultPrizeCategories> ,  
<defaultPrizeCategories> ,  
<defaultPrizeCategories> ,  
<defaultPrizeCategories> ],`

`<minPrizeCategories> ::= { <minimumID> <divident> <winners>  
<distributed> <jackpot> <fixed>  
<categoryType> <minGameType>  
<minimumDistributed> },`

`<defaultPrizeCategories> ::= { <id> <divident> <winners>  
<distributed> <jackpot> <fixed>`

```

        <categoryType> <gameType> }
    | { <id> <divident> <winners>
        <distributed> <jackpot> <fixed>
        <categoryType> <gameType> },
        <defaultPrizeCategories>

<wagerStatistics> ::= wagerStatistics : { <columns>
        <wagers> <addOn> }

<sort> ::= sort : [ { <direction> <property> <ignoreCase>
        <nullHandling> <descending> <ascending> } ],

<last> ::= last : { <gameId> <drawId> <drawTime>
        <status> <drawBreak> <visualDraw>
        <PricePoints> <winningNumbers>
        <prizeCategories> <wagerStatistics> },

<active> ::= active : { <gameId> <drawId> <drawTime>
        <status> <drawBreak> <visualDraw>
        <pricePoints> <prizeCategories>
        <wagerStatistics> },

<content> ::= content : [ <contentLoop> ], <totalPages>
    
```

<totalElements> <lastBoolean>

<numberOfElements> <sort> <first> <size>

<number>

<contentLoop> ::= { <gameId> <drawId> <drawTime>  
<status> <drawBreak> <visualDraw>  
<PricePoints> <winningNumbers>  
<prizeCategories> <wagerStatistics> },  
<contentLoop>  
| { <gameId> <drawId> <drawTime>  
<status> <drawBreak> <visualDraw>  
<PricePoints> <winningNumbers>  
<prizeCategories> <wagerStatistics> }

## Βασικές συναρτήσεις

(αφορούν τα παραπάνω)

<arrayOfList> ::= <integer> ,  
<integer> ,  
<integer> ,  
<integer> ,  
<integer>

<jsonElement> ::= <identifier> , <jsonElement>

| `<integer>` , `<jsonElement>`

| `<float>` , `<jsonElement>`

| `<real>` , `<jsonElement>`

| `<identifier>`

| `<integer>`

| `<float>`

| `<real>`

| `<digit>`

Αρχική συνάρτηση *(η πρώτη συνάρτηση που περιέχει τις παραπάνω)*

`<start>`        `::=`    { `<last>` `<active>` }  
                     | { `<content>` }

## b) Έλεγχος στοιχείων “last” &amp; “active”

Παράδειγμα επιτυχούς ανάλυσης

Εκτελώντας το αρχείο **last\_result.json** στη σωστή του μορφή, ο parser θα εμφάνιζε:

```
VAGGELIS@KVLOBOURO-PC /cygdrive/c/Compiler with Cygwin
$ ./myParser.exe last_result.json
~
~
~
Parsed successfully!
~
~
~
```

Παράδειγμα ανεπιτυχούς ανάλυσης

Στο ενδεχόμενο όπου το αρχείο **last\_result.json** εμφάνιζε πρώτα το στοιχείο “active” και έπειτα το “last”, ο συντακτικός αναλυτής θα εντόπιζε άμεσα το πρόβλημα και θα καταλήγαμε με το αποτέλεσμα:

```
VAGGELIS@KVLOBOURO-PC /cygdrive/c/Compiler with Cygwin
$ ./myParser.exe last_result.json
!!!
Parser error in line 2.
!!!
```

Για JSON input:

```
{ } last_result.json > { } active > [ ] prizeCategories > { } 0
1   {
2   |   "active": {
3   |       "gameId": 5104,
4   |       "drawId": 2391,
5   |       "drawTime": 1642536000000,
```

...



## Ερώτημα 2

Ομοίως με τη δομή του ερωτήματος 1, έγιναν προσθήκες στα αρχεία flex / bison, ώστε να υποστηρίζονται τα νέα στοιχεία και η δομή που ισχύει για το αρχείο **range\_result.json**.

### Παράδειγμα επιτυχούς ανάλυσης

Εκτελώντας το αρχείο **range\_result.json** στη σωστή του μορφή, ο parser θα εμφάνιζε:

```
VAGGELIS@KVELOBOURO-PC /cygdrive/c/Compiler with Cygwin
$ ./myParser.exe last_result.json
~*~*~
Parsed successfully!
~*~*~
```

### Παράδειγμα ανεπιτυχούς ανάλυσης

Στην περίπτωση που το αρχείο **range\_result.json** περιείχε αριθμό στο στοιχείο “size”, τη στιγμή που έχει οριστεί ως τύπου θετικού ακεραίου, ο συντακτικός αναλυτής εντοπίζει το πρόβλημα και θα καταλήγαμε με το αποτέλεσμα:

```
VAGGELIS@KVELOBOURO-PC /cygdrive/c/Compiler with Cygwin
$ ./myParser.exe range_result.json
!!!
Parser error in line 427.
!!!
```

Για JSON input:

```

424         ],
425     ],
426     "first":true,
427     "size":"10",
428     "number":0
429 }

```

## Ερώτημα 3

Υποερώτημα a

-

Υποερώτημα b

Αλλάζοντας τη συνάρτηση για το στοιχείο **"prizeCategories"**, ορίζουμε ως το 1<sup>ο</sup> nested JSON αντικείμενο αυτό με το **"id" = 1** και ακολουθούν ακριβώς 7 nested JSON με τα υπόλοιπα **"id"**.

```

// precisely 1 minimum + 7 default
prizeCategories:
    T_PRIZECATEGORIES T_COLON T_LBRACKET minPrizeCategories
    defaultPrizeCategories T_COMMA
    defaultPrizeCategories T_COMMA
    defaultPrizeCategories T_COMMA
    defaultPrizeCategories T_COMMA
    defaultPrizeCategories T_COMMA
    defaultPrizeCategories T_COMMA
    defaultPrizeCategories
    T_RBRACKET T_COMMA
;

```

Υποερώτημα c

Ο πίνακας που ορίζεται για το στοιχείο **"list"**, περιέχει αυστηρά 5 ακραίους. Το εύρος 1-45, δυστυχώς, δεν ελέγχεται.

```

arrayOfList:
    integer T_COMMA
    integer T_COMMA
    integer T_COMMA

```

```
integer T_COMMA
integer
;
```

## Παραδοχές – Σχόλια – Οδηγίες

- Από τα ερωτήματα 1 και 2 εμφανίζονται ως κύρια στοιχεία τα **“last”**, **“active”** και **“content”**. Βάσει αυτών εκτελούνται όλες οι συναρτήσεις του συντακτικού αναλυτή, αλλά στην αρχή ορίσαμε ως εναρκτήρια συνάρτηση μία δική μας, με όνομα *start*.
- Ορίσαμε δύο ειδών Boolean RegEx. Το *boolean* περιέχει τιμές **“0/1”**, ενώ το *boolean1* **“true/false”**.
- Ορίσαμε την RegEx property ομοίως με την RegEx identifier, με τη διαφορά ότι το πρώτο ακολουθεί τη *σύνταξη property κλάσης*.
- Εντός του bison αρχείου ορίζεται μια επιπλέον συνάρτηση *integer*, η οποία περιέχει μεν το *integer token* από το flex αρχείο, αλλά επίσης περιέχει και μικρότερα *integer tokens* (πχ το *minID*), προκειμένου να λυθούν προβλήματα κατά το parsing, λόγω διπλοτυπιών.
- Ομοίως ορίσαμε και *boolean* συνάρτηση.
- Το στοιχείο **“list”**, εντός του **“winningNumbers”**, περιέχει τους αριθμούς αυστηρά χωρισμένους με τουλάχιστον έναν κενό χαρακτήρα μετά από κάθε κόμμα.

Για parsing εκετλούμε τις παρακάτω εντολές:

- `bison -y -d myParser.y`
- `flex myParser.l`
- `gcc -c y.tab.c lex.yy.c`
- `gcc y.tab.o lex.yy.o -o myParser`
- `./myParser.exe <όνομα_αρχείου>`

## Κώδικας

Flex

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "y.tab.h"
%}

%option noyywrap
%option yylineno

/* regular expressions */

digit      [0-9]
property   (["] [a-zA-Z .] [a-zA-Z0-9 .]* ["])
minID      [1]
IDdigit    [2-8]
boolean    [0-1]
boolean1    true|false
integer     "0" | {digit}+
quotation  ["]
identifier (["] [a-zA-Z] [a-zA-Z0-9]* ["])
whitespace [ \t\v\n\f]
float       ({digit}+.{digit}+)
real        ([-]{float}|[-]{integer})

%%

"\last\""           {return T_LAST;}
"\active\""         {return T_ACTIVE;}
"\content\""        {return T_CONTENT;}

"\gameId\""         {return T_GAMEID;}
"\drawId\""         {return T_DRAWID;}
"\drawTime\""       {return T_DRAWTIME;}
"\status\""         {return T_STATUS;}
"\drawBreak\""      {return T_DRAWBREAK;}
"\visualDraw\""     {return T_VISUALDRAW;}
"\pricePoints\""    {return T_PRICEPOINTS;}
"\amount\""         {return T_AMOUNT;}
"\winningNumbers\"" {return T_WINNINGNUMBERS;}
"\prizeCategories\"" {return T_PRIZECATEGORIES;}
```

```

"\wagerStatistics\"      {return T_WAGERSTATISTICS;}

"\list\"                  {return T_LIST;}
"\bonus\"                 {return T_BONUS;}

"\id\"                    {return T_ID;}
"\divident\"              {return T_DIVIDENT;}
"\winners\"               {return T_WINNERS;}
"\distributed\"           {return T_DISTRIBUTED;}
"\jackpot\"               {return T_JACKPOT;}
"\fixed\"                 {return T_FIXED;}
"\categoryType\"          {return T_CATEGORYTYPE;}
"\gameType\"              {return T_GAMETYPE;}
"\minimumDistributed\"    {return T_MINIMUMDISTRIBUTED;}

"\columns\"               {return T_COLUMNS;}
"\wagers\"                {return T_WAGERS;}
"\addOn\"                 {return T_ADDON;}

"\totalPages\"            {return T_TOTALPAGES;}
"\totalElements\"         {return T_TOTALELEMENTS;}

"\first\"                 {return T_FIRST;}
"\numberOfElements\"      {return T_NUMBEROFELEMENTS;}
"\sort\"                  {return T_SORT;}
"\size\"                  {return T_SIZE;}
"\number\"                {return T_NUMBER;}
"\direction\"             {return T_DIRECTION;}
"\property\"              {return T_PROPERTY;}
"\ignoreCase\"            {return T_IGNORECASE;}
"\nullHandling\"          {return T_NULLHANDLING;}
"\descending\"            {return T_DESCENDING;}
"\ascending\"             {return T_ASCENDING;}

"{"                       {return T_LCURLY;}
"}"                       {return T_RCURLY;}
","                       {return T_COMMA;}
":"                       {return T_COLON;}
"["                       {return T_LBRACKET;}
"]"                       {return T_RBRACKET;}

[ \t\n\v\b]              ;      /* skip whitespace */

{minID}                   {return T_MINID;}
{IDdigit}                 {return T_IDDIGIT;}
{boolean}                 {return T_BOOLEAN;}
{boolean1}                {return T_BOOLEAN1;}
{integer}                 {return T_INTEGER;}

```

```

{quotation}          {return T_Q;}
{identifier}          {
                        yyldata.str = strdup(yytext);
                        return T_IDENTIFIER;
                      }

{property}            {
                        yyldata.str = strdup(yytext);
                        return T_PROPERTY1;
                      }

{float}               {return T_FLOAT;}
{real}                {return T_REAL;}

{whitespace}          {/*ignore whitespace*/}
%%

```

## Bison

```

%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

extern int yylex();
extern int yyparse();
extern FILE *yyin;
extern FILE *yyout;
extern int yylineno;
extern int yywrap;

int yylex();
void yyerror(const char* s);
%}

%union {
char* str;
}

%token T_Q
%token T_CONTENT
%token T_CONTENTLOOP
%token T_LAST
%token T_COLON
%token T_INTEGER

```

```
%token T_IDENTIFIER
%token T_FLOAT
%token T_GAMEID
%token T_DRAWID
%token T_DRAWTIME
%token T_STATUS
%token T_DRAWBREAK
%token T_VISUALDRAW
%token T_PRICEPOINTS
%token T_AMOUNT
%token T_WINNINGNUMBERS
%token T_PRIZECATEGORIES
%token T_WAGERSTATISTICS
%token T_LIST
%token T_BONUS
%token T_ID
%token T_DIVIDENT
%token T_WINNERS
%token T_DISTRIBUTED
%token T_JACKPOT
%token T_FIXED
%token T_CATEGORYTYPE
%token T_GAMETYPE
%token T_MINIMUMDISTRIBUTED
%token T_COLUMNS
%token T_WAGERS
%token T_ADDON

%token T_TOTALPAGES
%token T_TOTALELEMENTS
%token T_FIRST
%token T_NUMBEROFELEMENTS
%token T_SORT
%token T_SIZE
%token T_NUMBER
%token T_DIRECTION
%token T_PROPERTY
%token T_IGNORECASE
%token T_NULLHANDLING
%token T_DESCENDING
%token T_ASCENDING
%token T_BOOLEAN1
%token T_PROPERTY1

%token T_COMMA
%token T_LBRACKET
%token T_RBRACKET
%token T_LCURLY
```

```

%token T_RCURLY
%token T_ACTIVE
%token T_IDDIGIT
%token T_BOOLEAN
%token T_MINID
%token T_DIGIT
%token T_REAL

%%

// JSON initialization
start:
    T_LCURLY last active T_RCURLY    {printf("\t~~~\t\t\nParsed
successfully!\n\t~~~");}
    | T_LCURLY content T_RCURLY    {printf("\t~~~\t\t\nParsed
successfully!\n\t~~~");}
    ;

// main tokens
last:
    T_LAST T_COLON T_LCURLY gameId drawId drawTime status drawBreak
    visualDraw pricePoints winningNumbers prizeCategories
wagerStatistics
    T_RCURLY T_COMMA
    ;
active:
    T_ACTIVE T_COLON T_LCURLY gameId drawId drawTime status
    drawBreak visualDraw pricePoints prizeCategories wagerStatistics
T_RCURLY
    ;
contentLoop:
    T_LCURLY gameId drawId drawTime status drawBreak
    visualDraw pricePoints winningNumbers prizeCategories
    wagerStatistics T_RCURLY T_COMMA contentLoop
    | T_LCURLY gameId drawId drawTime status drawBreak
    visualDraw pricePoints winningNumbers prizeCategories
    wagerStatistics T_RCURLY
    ;

content:
    T_CONTENT T_COLON T_LBRACKET contentLoop T_RBRACKET
T_COMMA totalPages
    totalElements lastBoolean numberOfElements sort first size number
    ;

// separately declared Integer function
integer:
    T_INTEGER

```



```

| T_MINID
| T_IDDIGIT
| T_BOOLEAN
;

// separately declared boolean function
boolean:
    T_MINID
    | T_BOOLEAN
;

// nested Declarations
gameId:
    T_GAMEID T_COLON integer T_COMMA
;

drawId:
    T_DRAWID T_COLON integer T_COMMA
;

drawTime:
    T_DRAWTIME T_COLON integer T_COMMA
;

status:
    T_STATUS T_COLON T_IDENTIFIER T_COMMA
    | T_STATUS T_COLON T_ACTIVE T_COMMA
;

drawBreak:
    T_DRAWBREAK T_COLON integer T_COMMA
;

visualDraw:
    T_VISUALDRAW T_COLON integer T_COMMA
;

pricePoints:
    T_PRICEPOINTS T_COLON T_LCURLY T_AMOUNT T_COLON T_FLOAT T_RCURLY
T_COMMA
;

winningNumbers:
    T_WINNINGNUMBERS T_COLON T_LCURLY list bonus T_RCURLY T_COMMA
;

bonus:

```

```

T_BONUS T_COLON T_LBRACKET integer T_RBRACKET
;

// precisely 1 minimum + 7 default
prizeCategories:
  T_PRIZECATEGORIES T_COLON T_LBRACKET minPrizeCategories
  defaultPrizeCategories T_COMMA
  defaultPrizeCategories T_COMMA
  defaultPrizeCategories T_COMMA
  defaultPrizeCategories T_COMMA
  defaultPrizeCategories T_COMMA
  defaultPrizeCategories T_COMMA
  defaultPrizeCategories
  T_RBRACKET T_COMMA
;

// id = 1 strictly
minPrizeCategories:
  T_LCURLY minID dividant winners distributed jackpot fixed
  categoryType minGameType minimumDistributed T_RCURLY T_COMMA
;

// id 2-8
defaultPrizeCategories:
  T_LCURLY id dividant winners distributed jackpot fixed
  categoryType gameType T_RCURLY
;

id:
  T_ID T_COLON T_IDDIGIT T_COMMA
;

// Special case for id = 1
// minimumDistributed is also included
minID:
  T_ID T_COLON T_MINID T_COMMA
;

// Special case for id = 1
// minimumDistributed is also included
minGameType:
  T_GAMETYPE T_COLON T_IDENTIFIER T_COMMA
;

divident:
  T_DIVIDENT T_COLON T_FLOAT T_COMMA
;

```

```

winners:
    T_WINNERS T_COLON integer T_COMMA
    ;

distributed:
    T_DISTRIBUTED T_COLON T_FLOAT T_COMMA
    ;

jackpot:
    T_JACKPOT T_COLON T_FLOAT T_COMMA
    ;

fixed:
    T_FIXED T_COLON T_FLOAT T_COMMA
    ;

categoryType:
    T_CATEGORYTYPE T_COLON boolean T_COMMA
    ;

gameType:
    T_GAMETYPE T_COLON T_IDENTIFIER
    ;

minimumDistributed:
    T_MINIMUMDISTRIBUTED T_COLON T_FLOAT
    ;

wagerStatistics:
    T_WAGERSTATISTICS T_COLON T_LCURLY columns wagers addOn T_RCURLY
    ;

columns:
    T_COLUMNS T_COLON integer T_COMMA
    ;

wagers:
    T_WAGERS T_COLON integer T_COMMA
    ;

addOn:
    T_ADDON T_COLON T_LBRACKET jsonElement T_RBRACKET
    | T_ADDON T_COLON T_LBRACKET emptyJson T_RBRACKET
    ;

// exclusively empty JSON array
emptyJson:

```

```

;

list:
  T_LIST T_COLON T_LBRACKET arrayOfList T_RBRACKET T_COMMA
;
// accepted format for JSON int-array
arrayOfList:
  integer T_COMMA
  integer T_COMMA
  integer T_COMMA
  integer T_COMMA
  integer
;

jsonElement:
  T_IDENTIFIER T_COMMA jsonElement
| integer T_COMMA jsonElement
| T_FLOAT T_COMMA jsonElement
| T_REAL T_COMMA jsonElement
| T_IDENTIFIER
| integer
| T_FLOAT
| T_REAL
| T_DIGIT
;

totalPages:
  T_TOTALPAGES T_COLON integer T_COMMA
;

totalElements:
  T_TOTALELEMENTS T_COLON integer T_COMMA
;

lastBoolean:
  T_LAST T_COLON T_BOOLEAN1 T_COMMA
;

numberOfElements:
  T_NUMBEROFELEMENTS T_COLON integer T_COMMA
;

number:
  T_NUMBER T_COLON integer
;

size:
  T_SIZE T_COLON integer T_COMMA
;

```

```

first:
    T_FIRST T_COLON T_BOOLEAN1 T_COMMA
    ;

sort:
    T_SORT T_COLON T_LBRACKET T_LCURLY direction
    property ignoreCase nullHandling descending
    ascending T_RCURLY T_RBRACKET T_COMMA
    ;

direction:
    T_DIRECTION T_COLON T_IDENTIFIER T_COMMA
    ;

property:
    T_PROPERTY T_COLON T_PROPERTY1 T_COMMA
    ;

ignoreCase:
    T_IGNORECASE T_COLON T_BOOLEAN1 T_COMMA
    ;

nullHandling:
    T_NULLHANDLING T_COLON T_IDENTIFIER T_COMMA
    ;

descending:
    T_DESCENDING T_COLON T_BOOLEAN1 T_COMMA
    ;

ascending:
    T_ASCENDING T_COLON T_BOOLEAN1
    ;

%%

// Find the faulty line
void yyerror(const char* s)
{
    fprintf(stderr, "\t!!!\nParser error in line %d.\n\t!!!", yylineno);
    exit(1);
}

// Main function
int main (int argc, char **argv) {

    yyin = fopen(argv[1], "r"); // read permission
    yyparse ();

    return 0;
}

```

