Principles of Programming Languages & Translators

Department of Computer & Informatics Engineering

University of Patras

Spring Semester 2022

Teachers: I. Garofalakis, S. Sioutas, P. Hatzidoukas

Laboratory Exercise

The purpose of the lab is to familiarize yourself with describing a language in BNF format, basic concepts of compilers, and finally implement two of the parts of a compiler, a compiler and a parser using Bison and Flex tools.

PROCEDURES

Deliverable

ÿ **Written Report** that includes:

- o The details of the team members (Name, ID, Year, e-mail)
- o The description of the grammar of the language in BNF.
- o The **FINAL** language description files, which are given as **input** to Flex and Bison
- o Screenshots of application examples (at least one example for each query, for successful and unsuccessful analysis)
- o Comments - Assumptions made in developing the work

ÿ Compressed in **a file (zip)** the:

- o The above written reference.
- o The **FINAL** input files for flex and bison.
- o The **FINAL** code in C (and the .h file) produced by the two programs and the executable code of the parser.
- o The test files given as input to the parser to check for correctness its operation.

The zip file** must be named the registration numbers of the group members separated by _ and from the smallest to the largest (eg the1000_1543_2788_3072.zip) and submitted (MANDATORY) **names , the year and the respective registration numbers of the team members, as well as the e-mail addresses of all team members

Clarifications

ÿ For the operation of Flex and Bison tools you can find information in eclass of the lesson.

ÿ The exercise will be done in groups of 1 to 4 people.

ÿ Its contribution to the final grade is 30%.

ÿ The exercise is mandatory and its grade must be at least 5.

ÿ The final delivery dates will be determined according to the dates of the written exams in June and September respectively. The date of the oral examination of the work will be determined accordingly.

ÿ Any questions or suggestions are posted **EXCLUSIVELY** in the "Discussions" area at course page in eclass

(https://eclass.upatras.gr/modules/forum/?course=CEID1091)

EXERCISE DESCRIPTION

JSON is an open standard that uses human-readable text to convey informational data objects. Short for JavaScript Object Notation, it is a text format based on a subset of the JavaScript programming language, but is completely independent of any programming language. It is used as an alternative to XML for passing data between the server and web applications.

JSON consists of two structures. The first structure is a collection of name/value pairs, in most programming languages this can be thought of as an object, a structure a dictionary or a list of keys.

The second structure is an ordered list of values, in most programming languages this can be thought of as an array, list or sequence.

The above in JSON takes the following forms:

- **JSON String**
  It is a collection of 0 or more UNICODE characters, enclosed in double quotes "...", using the \ escape character.
  Examples from JSON Strings:
  - ""
  - "test"
  - "info \"within quotes\""
- **JSON Number**
  They are similar to C or Java numbers, except in octal and hexadecimal form. Additionally, the double quotes are not needed.
  Examples from JSON Numbers:
  - 20 (integer)
  - -20.50 (actual)
  - 5.3e-3 (scientific notation)
- **JSON Array**
  It is a collection of consecutive values enclosed in [ ] and separated by , (comma).

  Examples from JSON Arrays:
  - Array of Strings: ["one", "two", "three"]
  - Table of Numbers: [50, -12, 0.75]
  - Array with different types: [15, "string", -12.34]

Web services are a technology that allows applications to communicate with each other regardless of platform and programming language. A web service is a software interface that includes a collection of functions that can be accessed from the network through standard messages to describe an operation to be performed and data to be exchanged with another application.

OPAP SA provides the following online services through REST (Representational State Transfer) web services, in order to be used by any interested party. The relevant data is returned in JSON format.

For the use of web services through the url **https://www.opap.gr/web-services** you will not need to create an account or enter any API KEY. The Swagger file for Web Services can be found at: https://api.opap.gr/numerics/v1.0/api-swagger

The following methods are provided for KINO, POWERSPIN, SUPER3, PROTO, LOTTO, JOKER and EXTRA5 games:

Description	Call
Returns the DrawID, details, and results of the game's most recent draw/contest	https://api.opap.gr/draws/v3.0/{gameId}/last-result-and-active
Returns game sweepstakes/contests for a specified date range	https://api.opap.gr/draws/v3.0/{gameId}/draw-date/{fromDate}/{toDate}
Returns the latest contests/ draws for the game	https://api.opap.gr/draws/v3.0/{gameId}/last/{limit}
Returns the game's DrawIds for a given date range	https://api.opap.gr/draws/v3.0/{gameId}/draw-date/{fromDate}/{toDate}/draw-id
Returns the details and results of a draw or game contest for a given DrawId	https://api.opap.gr/draws/v3.0/{gameId}/{drawId}
Returns the game results/draw for a specified range Draw Ids	https://api.opap.gr/draws/v3.0/{gameId}/draw-id/{fromDrawId}/{toDrawId}
Returns the upcoming draw of the game	https://api.opap.gr/draws/v3.0/{gameId}/upcoming/1
Returns the active game contest	https://api.opap.gr/draws/v3.0/{gameId}/active
Returns game statistics (for all games except KINO)	https://api.opap.gr/games/v1.0/{gameId}/statistics
Returns game statistics for KINO	https://api.opap.gr/games/v1.0/1100/statisticks?drawRange=1801

Notes	
{gameId} - Game 1100 Kino 1110 Powerspin 2100 Super3 2101 Proto 5103 Lotto 5104 Tzoker 5106 Extra5	{fromDate} / {toDate} : Date format: yyyy-mm-dd (eg 2018-06-01 for June 1, 2018)
	{limit} : Number of last contests/ sweepstakes for which data will be returned
	{drawId} : Contest/draw ID

	{fromDrawId} / {toDrawId} : from – to range of DrawIDs of draws/contests to return data for

To get the results of the latest draw of the JOKER game (Gameid:5104 : Tzoker), we will use the following call, which returns the results in JSON format:

https://api.opap.gr/draws/v3.0/5104/last-result-and-active

Results may vary depending on when the call is made, as they always refer to the last draw. In the accompanying file **last_result.json** you will find an example of returned results for this call, based on which the language specifications are given below.

Language specifications

- Unless otherwise stated, whitespaces are not part of the language and play no role in syntax.
- The **"last"** element should appear before the **"active" element.**
- The **"last"** element should contain the following elements:
  - "gameId" - positive integer type
  - "drawId" - positive integer type
  - "drawTime" - positive integer type
  - "status" - alphanumeric type
  - "drawBreak" - positive integer type
  - "visualDraw" - positive integer type
  - "pricePoints" - this element should contain the element amount - of positive real type
  - "winningNumbers" - see below for details
  - "prizeCategories" - see details below
  - "wagerStatistics" - see below for details
- The "winningNumbers" element (inside the **"last" element)** should contain the following elements:
  - "list" - array type, which contains positive integers
  - "bonus" - array type, which contains a positive integer
- The **"active"** element should contain the following elements:
  - "gameId" - positive integer type
  - "drawId" - positive integer type
  - "drawTime" - positive integer type
  - "status" - alphanumeric type
  - "drawBreak" - positive integer type
  - "visualDraw" - positive integer type
  - "pricePoints" - this element should contain the element amount - of positive real type
  - "prizeCategories" - see details below
  - "wagerStatistics" - see below for details
- The "prizeCategories" element (either contained in the **"last"** element or in the **"active" element)** is of array type, which will contain nested JSON objects, which should have the following elements:
  - "id" - of type positive integer, in the range 1 - 8
  - "dividend" - positive real type
  - "winners" - positive integer type

- "distributed" - positive real type
- "jackpot" - positive real type
- "fixed" - of positive real type
- "categoryType" - of positive integer type 0 or 1
- "gameType" - alphanumeric type
- "minimumDistributed" - of type positive real. The particular item only appears when "id":1.

• The "wagerStatistics" element (whether contained in the **"last"** element or the **"active" element)** should contain the following elements:

- "columns" - positive integer type
- "wagers" - of positive integer type
- "addOn" - JSON Array type

Questions

1. (60%)

a. Give BNF the syntactic definition of the grammar of the language.

b. Using the Flex and Bison programs, implement a dictionary and parser, which will take as input a file written in the pseudo-language described above and check in one pass whether the program is syntactically correct. Your program will be called from the command line like this:

prompt> **myParser.exe file**

and will return the program itself to the screen and a diagnostic message about whether it was written correctly, or an appropriate error message (the line where the error is should be shown).

2. (30%)

Extend the above parser with the following functions:

The parser should be able to identify data that may have been returned from either the above call or the

https://api.opap.gr/draws/v3.0/5104/draw-date/{fromDate}/{toDate}

where the fromDate and toDate elements indicate the date range for which data has been requested and are in the format YYYY-MM-DD, where Y: Year, M: Month, D: Day. In the accompanying file **range_result.json** you can find an example of returned results for this call, based on which the additional language specifications are given below.

Therefore, the same parser should recognize test files whose content follows either the format of last_result.json or range_result.json (and not a combination of these).

Additional language specifications

- For the following elements, the display order should apply: **“content”, “totalPages”, “totalElements”, “last”, “numberOfElements”, “sort”, “first”, “size”, “number” .**

- The **“content”** element is of array type, which contains nested JSON objects that contain the following elements:
  - “gameId” - positive integer type
  - “drawId” - positive integer type
  - “drawTime” - positive integer type
  - “status” - alphanumeric type
  - “drawBreak” - positive integer type
  - “visualDraw” - positive integer type
  - “pricePoints” - this element should contain the element amount - of positive real type
  - “winningNumbers” - as defined in the 1st query
  - “prizeCategories” - as defined in the 1st query
  - “wagerStatistics” - as defined in the 1st query
- The **“totalPages”** element is of type positive integer.
- The **“totalElements”** element is of type positive integer.
- The element **“last”** is of type Boolean, with possible values true, false.
- The **“numberOfElements”** element is of type positive integer.
- The **“sort”** element is of array type, which contains an embedded JSON object with the following elements:
  - “direction” - alphanumeric type
  - “property” - alphanumeric type (property class syntax)
  - “ignoreCase” - type Boolean, with possible values true, false
  - “nullHandling” - alphanumeric type
  - “descending” - Boolean type, with possible values true, false
  - “ascending” - Boolean type, with possible values true, false
- The element **“first”** is of type Boolean, with possible values true, false.
- The **“size”** element is of type positive integer.
- The element **“number”** is of type positive integer

3. (10%)
   Make the necessary changes to the above so that:
   a. check the correctness of the value of the “gameId” element. The valid values for the element are shown in the 2nd table above.
   b. element “prizeCategories” to contain exactly 8 nested JSONs objects.
   c. the "list" element of "winningNumbers" contains exactly 5 positive ones integers, in the range 1 - 45.

   Otherwise, the analysis process is interrupted with the appearance of a relevant error.

Note: As test files (in addition to the 2 given in the zip file) you can either create your own, which will cover the correct and incorrect syntax cases, or use the calls given above to

API of OPAP.