Test ideas for Bulgarian Personal Identity number (ЕГН)

Task:

Context

You are testing web form for government site. You need to make sure the **format** validation of the ЕГН (единен граждански номер) field works correct.

In a text file list:

- All the test ideas you can come up with to test format validation.
- Example data for every test idea
- Explanation, why would you test this - what information you hope to uncover?

Resources:

- https://bg.wikipedia.org/wiki/%D0%95%D0%B4%D0%B8%D0%BD%D0%B5%D0%BD_%D0%B3%D1%80%D0%B0%D0%B6%D0%B4%D0%B0%D0%BD%D1%81%D0%BA%D0%B8_%D0%BD%D0%BE%D0%BC%D0%B5%D1%80
- http://www.budnaera.com/2010-03/2010-11-24_03.html
- https://www.lex.bg/forum/viewtopic.php?t=38153 – the mystery of digits 7-8-9
- Useful app to generate valid EGN - https://georgi.unixsol.org/programs/egn.php

- There’s also ton of good information in the application above, if you read the code comments of the author.

Test ideas:

Before we run into ideas, we should realize the EGN is not simply a number, we shouldn’t treat it like any number, there’s certain logic behind the way EGN is formed.

Example:

9504010850 – has different groups

Group 1 – year, month, day of birth represented in format YYMMDD

Group 2 – location of birth and gender, indicated by the last digit

Last digit – control number

- Looking at the data like this we will identify a special case related to date – if a person is born 02.02.1905 and 02.02.2005 which will both look like this 000202 for their first 6 digits. That’s why citizens born after year 2000 have 40 added to the month’s digits. Meaning a person born in 2005 will have their first 6 digits look like this 054202. This is a special case that we will like to consider.
- Similarly – people born before 1900 have 20 added to months, now – we will almost certainly not have live representatives, but since it’s a format validation, it’s worth testing.
- Another relation that might make sense to test – day to month. Ex. Not all months are 30/31 days long, we have February which is 28, but it’s 29 if the year is leap? Should the format validation know anything about leap years? It’s worth testing it.
- Group 2 is interesting as it represents the region of birth and the gender of the person. According to certain resources, here’s how

digit #7 is defined:

```
$EGN_REGIONS["Благоевград"]      = 43;  /* от 000 до 043 */
   $EGN_REGIONS["Бургас"]          = 93;  /* от 044 до 093 */
   $EGN_REGIONS["Варна"]           = 139; /* от 094 до 139 */
   $EGN_REGIONS["Велико Търново"]    = 169; /* от 140 до 169 */
   $EGN_REGIONS["Видин"]           = 183; /* от 170 до 183 */
   $EGN_REGIONS["Враца"]           = 217; /* от 184 до 217 */
   $EGN_REGIONS["Габрово"]         = 233; /* от 218 до 233 */
   $EGN_REGIONS["Кърджали"]        = 281; /* от 234 до 281 */
   $EGN_REGIONS["Кюстендил"]       = 301; /* от 282 до 301 */
   $EGN_REGIONS["Ловеч"]           = 319; /* от 302 до 319 */
   $EGN_REGIONS["Монтана"]         = 341; /* от 320 до 341 */
   $EGN_REGIONS["Пазарджик"]       = 377; /* от 342 до 377 */
   $EGN_REGIONS["Перник"]          = 395; /* от 378 до 395 */
   $EGN_REGIONS["Плевен"]          = 435; /* от 396 до 435 */
   $EGN_REGIONS["Пловдив"]         = 501; /* от 436 до 501 */
   $EGN_REGIONS["Разград"]         = 527; /* от 502 до 527 */
   $EGN_REGIONS["Русе"]           = 555; /* от 528 до 555 */
   $EGN_REGIONS["Силистра"]        = 575; /* от 556 до 575 */
   $EGN_REGIONS["Сливен"]          = 601; /* от 576 до 601 */
   $EGN_REGIONS["Смолян"]          = 623; /* от 602 до 623 */
   $EGN_REGIONS["София - град"]      = 721; /* от 624 до 721 */
   $EGN_REGIONS["София - окръг"]     = 751; /* от 722 до 751 */
   $EGN_REGIONS["Стара Загора"]      = 789; /* от 752 до 789 */
   $EGN_REGIONS["Добрич (Толбухин)"] = 821; /* от 790 до 821 */
   $EGN_REGIONS["Търговище"]         = 843; /* от 822 до 843 */
   $EGN_REGIONS["Хасково"]          = 871; /* от 844 до 871 */
   $EGN_REGIONS["Шумен"]            = 903; /* от 872 до 903 */
   $EGN_REGIONS["Ямбол"]            = 925; /* от 904 до 925 */
   $EGN_REGIONS["Друг/Неизвестен"]   = 999; /* от 926 до 999
```

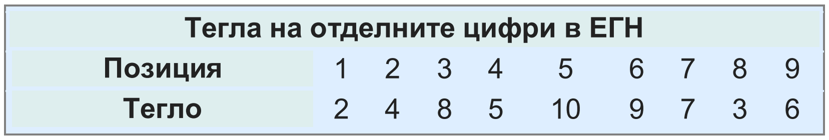
There's also information that the last region is not clearly defined – might be used for people born outside of Bulgaria, or when numbers for specific region were exceeded.

The region numbers are given based on statistical basis, therefore it happens that numbers for specific regions might be exceeded, so people might be given numbers "borrowed from other regions" or from the "other" region.

The 9-th digit is picked based on persons gender when it is even for boys and odd for girls.

From all these, we can conclude that it doesn't really make sense to test anything specific here, unless we have anything else than digits, because we don't have strict rules to apply here.

Last digit is control digit, it is used to make sure that the last digit belongs exactly to that number.



Тегла на отделните цифри в ЕГН									
Позиция	1	2	3	4	5	6	7	8	9
Тегло	2	4	8	5	10	9	7	3	6

We multiply digits on specific positions by their weights and divide the sum of all products by modulus 11. If the result is less than 10, it's taken as control digit. Otherwise, the control digit is zero.

This is common pattern to verify the validity of personal ID numbers, VAT numbers or any other fiscal identification numbers.

Test ideas:

- **8802021773** – to test regular case, just to make sure normal input is processed
- **0542150178** – person born after 2000. We can add a variation with the month being 10/11/12, so the 3rd digit is 5.
- **1642290088** – to test a person born in a valid leap year, I have suspicion that dev might forgot about leap years, or not considered them at all.
- **1542290037** – invalid EGN to test 29-th Feb in non-leap year
- **9502310099** – invalid date format 31 Feb, we can try other violations of these, for example 99 of March, just to make sure that validation is correct. Or 31 for month that's 30 days only.
- **9507310044** – to test 31 with months Jul and Aug, in case developer doesn't use a library or "predicts" whether the month is 31 or 30 by just taking turns.
- **6547310027** – date in the future is an interesting case, if format is concerned, this is not an error. It's true that it is not assigned, yet, but it's in the right format.
- **8802021775** – all correct, but the check digit is wrong, this way we can test if correct validation for forged numbers is performed.

This is as far as tickling with format validation is concerned. If we look at more generic tests, we can consider:

- Empty input – to verify correct error is displayed for empty input. Important – incorrect EGN won't be good error message, as there isn't input to be considered incorrect.
- All zeros, all nines, arbitrary numbers – all should produce errors. Failing to do so, will reveal serious lack of format validation.
- Less than 10 symbols, more than 10 symbols – both to produce error.
- Arbitrary injection of letters – to produce error.
- Arbitrary injection of special chars – it'd be a good idea if it's not possible at all. Normally, we expect the input to be limited to numbers only, if it is possible, this might lead to more in depth testing of XSS and injections possibilities.
- Spaces in beginning, middle, end – although it doesn't make sense to have spaces, normally we don't punish these in forms. We expect them to be trimmed or removed. It's normal to have space, if the user is copy-pasting it, for example.
- Anyway, if it is we can try various vulnerability attacks – SQL injection, XSS, server-side command injection, etc. All should produce errors.
- Consider testing of the client side and server side – best case scenario, we will have running validation on both ends.
- To test back end validation – we input invalid value as a parameter of the service call, not in the input field.
- We can run all the possible injection types against server as well.
- Consider testing in different browser type or version – Chrome, Opera, Edge, IE – this might reveal problems, especially of certain checks are heavy Javascript.
- Consider testing on mobile – is the field displayed correctly on smaller screen, in portrait and landscape orientation, is the appropriate keyboard displayed for inputting numbers, only.

- Trying to submit the field by clicking on the submit button, vs. submitting by hitting Enter. See if that works.