Contents

TECHNICAL TEST - 1.

Task

1. Function takes two and only two characters as input. It returns 1 when each entered symbol is either letter. or number and 0 in other cases.
2. Numbers are Hindu-Arabic numerals from 0 to 9. Letters are Latin, Cyrillic or Modern Greek alphabet.

The bug was found before, slogan is: "Cyrillic letter can be treated as non-letter "symbol".
Actions to reproduce:
Function input: АУ.
Expected output: 1. (Logic: only letters).
Observed output: 0.

3. The bug was fixed. Existing tests were already ran as a regression test after bugfix, here is the test report:

Input Expected output Status.

AA	1	Passed
1A	1	Passed
11	1	Passed
A$	0	Passed
(1	0	Passed

Actions and questions:

1. What tests cases will you do for bug's verification?
2. What tests cases will you propose to enforce regression testing of the function?
3. Negative tests like input less than 2 symbols and input more than 2 symbols are out of scope for both bug. verification and additional regression test cases.

Technical test review

Below are my notes and comments for 'TECHNICAL TEST - 1' after careful reading of the task description.
Here is a reference on some sections where I used numbers or section's name as the references. It will be more straightforward for readers.
Also, I created additional acceptance criteria based on the information from the task and added details why I need to extend the requirements.

TECHNICAL TEST - 1.

1. Function takes two and only two characters as input.
2. It returns 1 when each entered symbol is either letter or number and 0 in other cases.
3. Numbers are Hindu-Arabic numerals from 0 to 9.
   a) Q: What kind of "Expected output" should we have when entering only one number and any character (in total we have 2 characters?
4. Letters are Latin, Cyrillic or Modern Greek alphabet.

The bug was found before, slogan is: "*Cyrillic letter can be treated as non-letter symbol*"

Actions to reproduce:
Function input: АУ.
Expected output: 1. (Logic: only letters).
Observed output: 0.
The bug was fixed.

Existing tests were already ran as a regression test after bugfix, here is the test report:

Input	Expected output	Status
AA	1	Passed
1A	1	Passed
11	1	Passed
A$	0	Passed
(1	0	Passed

Additional questions for task writers:

1. What test cases will you do for bug's verification?
2. What test cases will you propose to enforce regression testing of the function?
3. Negative tests like input less than 2 symbols and input more than 2 symbols are out of scope for both bug verification and additional regression test cases.

Note:
We will be glad to see your reasoning and argumentation as an addition, but the main thing is to provide test cases for both bug verification and additional regression test cases.

Task execution actions

Below are my assumptions that can help to analyze the logic of the tester, the developer who implemented this code, and the business analyst who converted the user story into the task.

AC-1 Function takes two and only two characters as input.
AC-2 It returns 1 when each entered symbol is either letter or number and 0 in other cases.
AC-3 Numbers are Hindu-Arabic numerals from 0 to 9.
AC-4 Letters are Latin, Cyrillic or Modern Greek alphabet.

Question , for AC-2 clarification:

Q-1: What kind of “Expected output” should we have when entering only one number and any character (in total we have 2 characters for the input)?

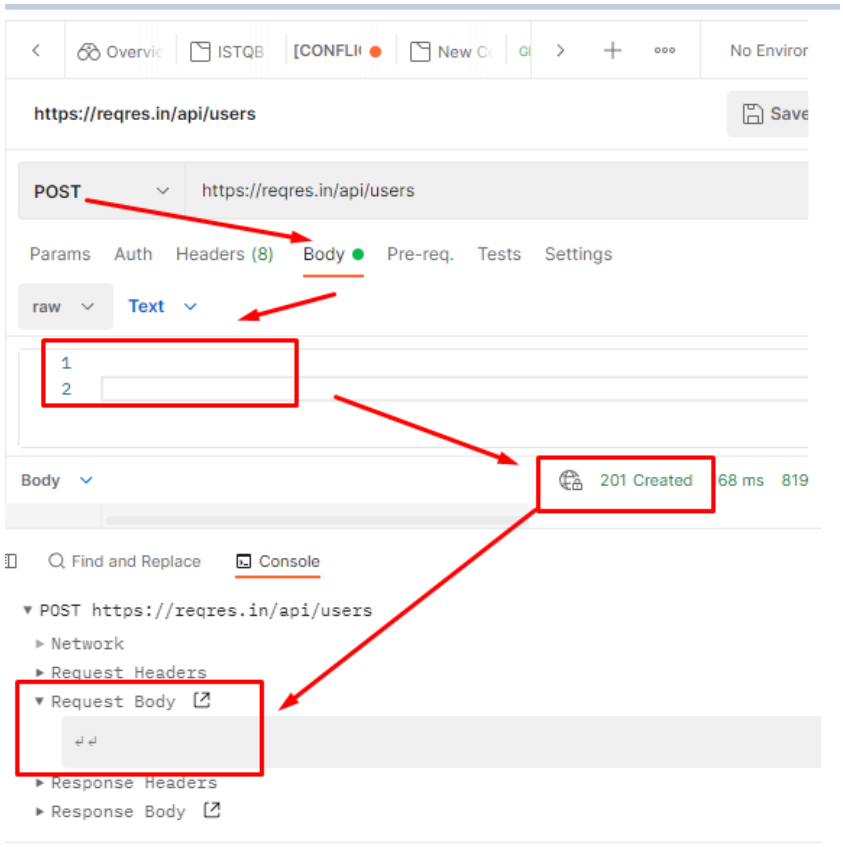
If “other cases” means that any combination of a single letter or number with a symbol will generate output = 0, please check “Table-1”.

Also here are a few more questions. Check them below.

Q-2: Information about the exact interface for validation is missing. I assume that we can test the logic using the web form (by browser) or mobile app or tablet app or desktop app or API or DB. In that case, the acceptance criteria should be updated for each case and describe all boundary values for validation.

Q-3: I assume that the task should be completed for API (REST) and in that case what kind of “OUTPUT” value we should get for non-printable characters, like white spaces, line breaks, etc?

I was checking the first free API and can assume that “INPUT” value can accept the same behavior. Please see the screenshot below.



If there is a missing requirement, I would suggest providing more information for the developers and then check implemented business logic.

Q4: Based on the bug and test report I can assume that input combinations with letters in lowercase were skipped. In that case "the lower case" should be included otherwise AC should have a precise description about this fact. My test data table contains all combinations, and this data should be attached when the next regression test will start. Also, here is my answer about enforcing tests and additional inputs.

By default, I checked requirements and always followed simple rules.

1. Task does not have options for unexpected behavior.
2. There are no ambiguities.
3. There are clear conditions for the input data.

4. There are clear conditions for the output data.
5. There are clear conditions for boundary values and cases.
6. There are clear descriptions for undetectable behavior.
7. There are clear descriptions for error messages (if it's applicable)

Below are all my answers that I found in your task.

Answers

Q1:
- my answer:

For the bug verification I will check "Output" using information that provided in the bug report and I will use test case with "Input" = 'АУ'

NOTE:

1. Based on the presented 'TECHNICAL ENTRY TEST' the bug report **<u>is not completed</u>**.

a) ***Expected output: 1. (Logic: only letters).*** is not acceptable in this case.
Based on AC-1, expected result should be updated with a phrase : ***Expected output: 1. (Logic: only letters or digits).***

b) Bug report is not clear for me and there are missing major sections, like steps to reproduce. I'm talking about the approved format of the bug report for the QA team but based on the presented information here it is not possible to understand how to execute actions related to input data and where the user / tester should see 'Observed output'.

c) Function input like 'AA' is not acceptable in this case. It is not possible to understand what kind of alphabet was used, Cyrillic or Latin. Bug report should be updated with exact information about the alphabet.

d) Comments like 'Bug was fixed' without any information about ENV, version of app, etc. is not acceptable. In case when QA team approved this flow I have skipped this information and have no questions.

Q2:
- my answer:

I will suggest extending the test data and will not create any additional test cases. There is no sense in creating a new TC based on the task description.

In my case I will create a new table of test data that will include all potential examples that the tester should check after the current bug fix. All information about test data I will describe later in Table - 1 / 2 / 3.

NOTE:

1. I assume that my update for AC, Q2 is wrong and it is not possible to use non-printable characters.
2. Otherwise, this topic should be discussed in the team or used any existing QA team approach for such cases.

Q3:

- my answer:

Negative test. As I mentioned early and based on existing AC all potential negative scenarios are not possible to image (you clearly described this in the '***Actions and questions' section, p.3***).

Here are a few words about negative scenarios that I prefer to use.

1. Provided information in the test report is not clear and as you can see for the 1st input - 'AA'. It is not possible to recognize the exact alphabet on "server side". I see here misleading information. If we are talking about 3 different alphabets, we can say that web form/API/DB will consume the data in different ASCII codes. Please, check the screen below.

ASCII - Binary Character Table

Letter	ASCII Code	Binary	Letter	ASCII Code	Binary
a	097	01100001	A	065	01000001
b	098	01100010	B	066	01000010
c	099	01100011	C	067	01000011

2. Even if we skipped this point of view, we can't ignore this fact when applying different test techniques like equivalence partitioning. We can't confirm without additional information that if we are testing 3 different alphabets, we have no rights to ignore information about the language in test cases.
3. If we have an API I would like to request information about 'bad requests' and will include this information into the test data table. Combination of non-printable characters should be tested as negative scenarios, also. How many 'inputs' we should have? I will say that it depends on the existing approach for testing 'inputs' and 'functions' for current features. At least one example should be checked as I described in the description for AC, Q3.
4. Basic negative scenario is to check the form behavior when data was not entered , but submitted.

Information below contains everything regarding extending regression tests. As you see I added tables of all letters for all alphabets. It helps analyze lists of letters, comparing visual views (here you will find my explanation why the bug report is not completed and should have information about the exact alphabet that was used during tests).

Also, I created 3 tables with test data that were filtered and final, 4rd table has a unique set of data for testing. 7th table contains the final list of values for 'INPUT' that should be attached into the test management system.

Created (existing) test case should have this list. I want to note, if we are talking about 'positive' test cases there is no reason for creating extra tests. It Is simpler to use a list of values (extend test data) instead of clone the same test. I have only one thing in my mind that can be used for dividing test cases by alphabet. In that case we can create 5 "different" test cases for unique alphabet, digits and special characters.

Table - 1. Acceptance criteria

Range #	Acceptable range of symbols	Total symbol amount	Expected output for entered symbols
1.	[0 - 9]	10	1
2.	Latin alphabet, letters [A - Z]	26	1
3.	Latin alphabet, letters [a-z]	26	1
4.	Modern Greek alphabet, letters [Α - Ω]	24	1
5.	Modern Greek alphabet, letters [α - ω]	24	1
6.	Cyrillic alphabet, letters [А - Я]	33	1
7.	Cyrillic alphabet, letters [а-я]	33	1
8.	Any special character (not a digit or letter)	????	0

Table - 2. ISO Latin alphabet

ISO Latin alphabet																									
Uppercase letter set	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Lowercase letter set	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z

Table - 3. ISO alphabet

ISO Cyrillic alphabet																														
Uppercase letter set	А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
Lowercase letter set	а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

Table - 4. ISO Modern Greek alphabet

ISO Modern Greek **alphabet**																								
Uppercase letter set	Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω
Lowercase letter set	α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο	π	ρ	σ/ς	τ	υ	φ	χ	ψ	ω

Table - 5. Input values and outputs (Basic data)

	1			2			3			4			5			6			7			8		
	[0 - 9], test character - 9			Latin alphabet, letters [A - Z], test character - Z			Latin alphabet, letters [a-z], test character - z			Modern Greek alphabet, letters [A - Ω], test character - Ω			Modern Greek alphabet, letters [α - ω], test character - ω			Cyrillic alphabet, letters [A - Я], test character - Я			Cyrillic alphabet, letters [а-я], test character - я			Any special character (not a digit or letter), test character -)		
	TC #	INPUT	OUTPUT	TC #	INPUT	OUTPUT	TC #	INPUT	OUTPUT	TC #	INPUT	OUTPUT	TC #	INPUT	OUTPUT	TC #	INPUT	OUTPUT	TC #	INPUT	OUTPUT	TC #	INPUT	OUTPUT
1	[0 - 9], test character - 0	**TC-1**	**09**	**1**	**TC-2**	**Z0**	**1**	**TC-3**	**z0**	**1**	**TC-4**	**Ω0**	**1**	**TC-5**	**ω0**	**1**	**TC-6**	**Я0**	**1**	**TC-7**	**я0**	**1**	**TC-8**	**)0**	0
2	Latin alphabet, letters [A - Z], test character - Z	TC-1-1	0Z	**1**	**TC-2 -1**	**ZZ**	**1**	**TC-3 -1**	**zZ**	**1**	**TC-4-1**	**ΩZ**	**1**	**TC-5-1**	**ωZ**	**1**	**TC-6-1**	**ЯZ**	**1**	**TC-7-1**	**яZ**	**1**	**TC-8-1**	**)Z**	0
3	Latin alphabet, letters [a-z], test character - z	TC-1-2	0z	**1**	TC-2 -2	Zz	**1**	**TC-3 -2**	**zz**	**1**	**TC-4-2**	**Ωz**	**1**	**TC-5-2**	**ωz**	**1**	**TC-6-2**	**Яz**	**1**	**TC-7-2**	**яz**	**1**	**TC-8-2**	**)z**	0
4	Modern Greek alphabet, letters [A - Ω], test character - Ω	TC-1-3	0Ω	**1**	TC-2 -3	ZΩ	**1**	TC-3-3	zΩ	**1**	**TC-4-3**	**ΩΩ**	**1**	**TC-5-3**	**ωΩ**	**1**	**TC-6-3**	**ЯΩ**	**1**	**TC-7-3**	**яΩ**	**1**	**TC-8-3**	**)Ω**	0
5	Modern Greek alphabet, letters [α - ω], test character - ω	TC-1-4	0ω	**1**	TC-2 -4	Zω	**1**	TC-3-4	zω	**1**	TC-4-4	Ωω	**1**	**TC-5-4**	**ωω**	**1**	**TC-6-4**	**Яω**	**1**	**TC-7-4**	**яω**	**1**	**TC-8-4**	**)ω**	0
6	Cyrillic alphabet, letters [A - Я], test character - Я	TC-1-5	0я	**1**	TC-2 -5	Zя	**1**	TC-3-5	zя	**1**	TC-4-5	Ωя	**1**	**TC-5-5**	ωя	**1**	**TC-6-5**	**Яя**	**1**	**TC-7-5**	**яя**	**1**	**TC-8-5**	**)я**	0
7	Cyrillic alphabet, letters [а-я], test character - я	TC-1-6	0Я	**1**	TC-2 -6	ZЯ	**1**	TC-3-6	zЯ	**1**	TC-4-6	ΩЯ	**1**	TC-5-6	ωЯ	**1**	TC-6-6	ЯЯ	**1**	**TC-7-6**	**яЯ**	**1**	**TC-8-6**	**)Я**	0

8	Any special character (not a number or letter), test character - $	TC-1-7	0$	0	TC-2-7	Z$	0	TC-3-7	z$	0	TC-4-7	Ω$	0	TC-5-7	ω$	0	TC-6-7	Я$	0	TC-7-7	я$	0	**TC-8-7**	)$	0

Table - 6. Input values and outputs (Filtered data)

		1			2			3			4			5			6			7			8		
		[0 - 9], test character - 9			Latin alphabet, letters [A - Z], test character - Z			Latin alphabet, letters [a-z], test character - z			Modern Greek alphabet, letters [A - Ω], test character - Ω			Modern Greek alphabet, letters [α - ω], test character - ω			Cyrillic alphabet, letters [А - Я], test character - Я			Cyrillic alphabet, letters [а-я], test character - я			Any special character (not a digit or letter), test character -)		
		TC #	INPUT	OUTPUT	TC #	INPUT	OUTPUT	TC #	INPUT	OUTPUT	TC #	INPUT	OUTPUT	TC #	INPUT	OUTPUT	TC #	INPUT	OUTPUT	TC #	INPUT	OUTPUT	TC #	INPUT	OUTPUT
1	[0 - 9], test character - 0	TC-1	09	1	TC-2	Z0	1	TC-3	z0	1	TC-4	Ω0	1	TC-5	ω0	1	TC-6	Я0	1	TC-7	я0	1	TC-8	)0	0
2	Latin alphabet, letters [A - Z], test character - Z	-	-	-	TC-2-1	ZZ	1	TC-3-1	zZ	1	TC-4-1	ΩZ	1	TC-5-1	ωZ	1	TC-6-1	ЯZ	1	TC-7-1	яZ	1	TC-8-1	)Z	0
3	Latin alphabet, letters [a-z], test character - z	-	-	-	-	-	-	TC-3-2	zz	1	TC-4-2	Ωz	1	TC-5-2	ωz	1	TC-6-2	Яz	1	TC-7-2	яz	1	TC-8-2	)z	0
4	Modern Greek alphabet, letters [A - Ω], test character - Ω	-	-	-	-	-	-	-	-	-	TC-4-3	ΩΩ	1	TC-5-3	ωΩ	1	TC-6-3	ЯΩ	1	TC-7-3	яΩ	1	TC-8-3	)Ω	0
5	Modern Greek alphabet, letters [α - ω], test character - ω	-	-	-	-	-	-	-	-	-	-	-	-	TC-5-4	ωω	1	TC-6-4	Яω	1	TC-7-4	яω	1	TC-8-4	)ω	0
6	Cyrillic alphabet, letters [А - Я], test character - Я	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	TC-6-5	Яя	1	TC-7-5	яя	1	TC-8-5	)я	0
7	Cyrillic alphabet, letters [а-я], test character - я	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	TC-7-6	яЯ	1	TC-8-6	)Я	0

8	Any special character (not a digit or letter), test character - $	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	TC-8-7	)$	0

Table - 7. Input values and outputs (Final data)

#	TC	INPUT	OUTPUT
1	TC-1	09	1
2	TC-2	Z0	1
3	TC-2-1	ZZ	1
4	TC-3	z0	1
5	TC-3-1	zZ	1
6	TC-3-2	zz	1
7	TC-4	Ω0	1
8	TC-4-1	ΩZ	1
9	TC-4-2	Ωz	1
10	TC-4-3	ΩΩ	1
11	TC-5	ω0	1
12	TC-5-1	ωZ	1

#	TC	INPUT	OUTPUT
13	TC-5-2	ωz	1
14	TC-5-3	ωΩ	1
15	TC-5-4	ωω	1
16	TC-6	Я0	1
17	TC-6-1	ЯZ	1
18	TC-6-2	Яz	1
19	TC-6-3	ЯΩ	1
20	TC-6-4	Яω	1
21	TC-6-5	Яя	1
22	TC-7	я0	1
23	TC-7-1	яZ	1
24	TC-7-2	яz	1

#	TC	INPUT	OUTPUT
25	TC-7-3	яΩ	1
26	TC-7-4	яω	1
27	TC-7-5	яя	1
28	TC-7-6	яЯ	1
29	TC-8	)0	0
30	TC-8-1	)Z	0
31	TC-8-2	)z	0
32	TC-8-3	)Ω	0
33	TC-8-4	)ω	0
34	TC-8-5	)я	0
35	TC-8-6	)Я	0
36	TC-8-7	)$	0