Equivalence Classes Partitioning

Identification of Equivalence Classes

Equivalence Classes will be derived from inputs and outputs. We will proceed one element at the time. The main principle is to identify *valid* and *invalid* partitions based on the understanding of the problem.

UserName

- **Valid ECs**
  - EC1: 6 <= UserName length <= 12
  - EC2: UserName starts with letter
  - EC3: UserName includes only letters and digits
- **Invalid ECs**
  - EC4: UserName empty
  - EC5: 1 <= UserName length < 6
  - EC6: UserName length > 12
  - EC7: UserName starts with non-letter character
  - EC8: UserName includes other characters than letters and digits

FirstName

- **Valid ECs**
  - EC9: FirstName empty
  - EC10: FirstName starts with letter
  - EC11: FirstName includes only letters and spaces
- **Invalid ECs**
  - EC12: FirstName starts with non-letter character
  - EC13: FirstName includes other characters than letters and spaces

LastName

- **Valid ECs**
  - EC14: LastName empty

- EC15: LastName starts with letter
- EC16: LastName includes only letters and spaces

- **Invalid ECs**
  - EC17: LastName starts with non-letter character
  - EC18: LastName includes other characters than letters and spaces

Age

- **Valid ECs**
  - EC19: 18 <= Age < 65
- **Invalid ECs**
  - EC20: Age empty
  - EC21: Age < 18
  - EC22: Age >= 65
  - EC23: Age not an integer number

Email

- **Valid ECs**
  - EC24: Email format is <*local-part*>@<*domain*>(1 @)
  - EC25: <*local-part*> includes only letters, digits and characters '.', '_', '+', '-'
  - EC26: number of fragments in <*domain*> >= 2
  - EC27: each fragment except last includes only letters, digits, '-'
  - EC28: last fragment includes only letters
  - EC29: 2 <= length of last fragment <= 6
- **Invalid ECs**
  - EC30: Email is empty
  - EC31: Email format is different to <*local-part*>@<*domain*>(no @)
  - EC32: Email format is different to <*local-part*>@<*domain*>(more than 1 @)
  - EC33: <*local-part*> includes other characters than letters, digits, '.', '_', '+', '-'
  - EC34: number of fragments in <*domain*> < *2*

- EC35: fragment (not last) includes other characters than letters, digits, '-'
- EC36: last fragment includes other characters than letters
- EC37: last fragment length < 2
- EC38: last fragment length > 6

City

- **Valid ECs**
  - EC39: City one of {Ottawa, Toronto, Montreal, Halifax}
- **Invalid ECs**
  - EC40: City is empty
  - EC41: City not one of {Ottawa, Toronto, Montreal, Halifax}

Postal Code

We refer to a character in a Postal Code at position *i* as C_i. A valid Postal Code could have a blank between C_3 and C_4.

- **Valid ECs**
  - EC42: Postal Code format is $C_1C_2C_3C_4C_5C_6$
  - EC43: Postal Code format is $C_1C_2C_3$ $C_4C_5C_6$ (with blank in middle)
  - EC44: C_1 one of *{A,B,C,E.G,H,J,K,L,M,N,P,R,S,T,V,X,Y}*
  - EC45: C_3, C_5 one of *{A,B,C,E.G,H,J,K,L,M,N,P,R,S,T,V,W,X,Y,Z}*
  - EC46: C_2, C_4, C_6 are digits
- ***Invalid ECs***
  - EC47: Postal Code length < 6
  - EC48: Postal Code length > 7
  - EC49: C_1 is one of {D,F,I,O,Q,U}
  - EC50: C_1 is one of {W,Z}
  - EC51: C_1 is a letter not capital
  - EC52: C_1 is not a letter
  - EC53: C_2 is not a digit

- EC54: C_3 is one of {D,F,I,O,Q,U}
- EC55: C_3 is a letter not capital
- EC56: C_3 is not a letter
- EC57: C_4 is not a digit
- EC58: C_5 is one of {D,F,I,O,Q,U}
- EC59: C_5 is a letter not capital
- EC60:C_5 character is not a letter
- EC61: C_6 is not a digit

Output (Result)

- ***Valid ECs***
  - EC62: registration request accepted
  - EC63: Err1
  - EC64: Err2
  - EC65: Err3
  - EC66: Err4
  - EC67: Err5
  - EC68: Err6
  - EC69: Err7
  - EC70: Err8
  - EC71: Err9
  - EC72: Err10
  - EC73: Err11

  Note that the error messages are considered as "valid" outputs as they are specified outputs in the specification. Having them listed ensures we generate test cases that check the right handling of errors. These test cases will however involve invalid inputs.
- ***Invalid ECs***
  - EC74: any other output

Boundary Value Analysis

We will identify boundary conditions based on the equivalence classes along with sample data. A boundary condition is a condition at the edge of an equivalence classes. For letters, we assume the ASCII code ordering where 'A' would have the lowest code number and 'z' the highest.

Input Field	ECs	Boundary Condition	Sample Values
UserName (valid)	EC1, EC2, EC3	length = 6, start with 'A', only letters and digits	A00000

Input Field	ECs	Boundary Condition	Sample Values
	EC1, EC2, EC3	length = 12, start with ‘z’, only letters and digits	zzzzzzzzzzzz
UserName (invalid)	EC4	empty	“”
	EC5	length = 1	z, Z, a, A
	EC6	length = 5	zzzzz
	EC7	start with non-letter	@adr278a, [vvvvvvvv, `8888888, {abcd123e, 0xyz234, 999zzzzz
	EC8	include non letters or digits	D@XYCD56A, bbbbbbbbbbb`
FirstName (valid)	EC9	empty	“”
	EC10, EC11	start with letter, only letters and spaces	A, “A ”, zzzzzzzzzzzzzzzzz
FirstName (invalid)	EC12	start with non-letter	@, “[”, `zzzzzzzzz
	EC13	include non letters	A@aaaaaaaaaa, “z abc xyz’”
LastName (valid)	EC14	empty	“”
	EC15, EC16	start with letter, only letters and spaces	A, “A ”, zzzzzzzzzzzzzzzzz
LastName (invalid)	EC17	start with non-letter	@, “[”, `zzzzzzzzz
	EC18	include non letters	A@aaaaaaaaaa, “z abc xyz’”
Age (valid)	EC19	Age = 18	18
	EC19	Age = 64	64
Age (invalid)	EC20	Age empty	“”
	EC21	Age < 18	17, 0
	EC22	Age >= 65	65
	EC23	not integer number	/, :, 1/
Email (valid)	EC24, EC25, EC26, EC27, EC28, EC29	format <*local-part*>@<*domain*>, <*local-part*> includes only letters, digits and characters ‘.’, ‘_’, ‘+’, ‘-’, 2 fragments in <*domain*>, last fragment 2 letters	+@-.AA,
	EC24, EC25, EC26,	format <*local-part*>@<*domain*>,	zzZZz.zz999@zzz.zzzz99999zzzzzzz.zzzzzzz9999zzzzz.zzzzzz

Input Field	ECs	Boundary Condition	Sample Values
	EC27, EC28, EC29	<*local-part*> includes only letters, digits and characters '.', '_', '+', '-', more than 2 fragments in <*domain*>, last fragment 6 letters	
Email (invalid)	EC30	empty	""
	EC31	no @	paul.john.somewhere.com
	EC32	more than one @	paul.john@some.where@else.com
	EC33	<*local-part*> includes other character than letters, digits, '.', '_', '+', '-'	paul*@somewhere.com
	EC34	1 fragment in <*domain*>	paul@com
	EC35	fragment (not last) includes other than letters, digits, '-'	paul@some{where.com
	EC36	last fragment includes other than letters	paul@somewhere.c[m
	EC37	last fragment length 1	paul@somewhere.c
	EC38	last fragment length 7	paul@somewhere.cooooom
City (valid)	EC39	city in {Ottawa, Toronto, Montreal, Halifax}	Halifax, Toronto
City (invalid)	E40	city empty	""
	E41	city not in {Ottawa, Toronto, Montreal, Halifax}	Vancouver, Calgary
Postal Code (valid)	EC42, EC43, EC44, EC45, EC46	format is $C_1C_2C_3C_4C_5C_6$, characters C_1 ,C_3 , C_5 are 'A', characters C_2 ,C_4 , C_6 are '0'	A0A0A0
	EC42, EC43, EC44,	format is $C_1C_2C_3C_4C_5C_6$, characters C_1 is 'Y', C_3 , C_5 are	Y9Z9Z9

Input Field	ECs	Boundary Condition	Sample Values
	EC45, EC46	'Z', characters C_2 ,C_4 , C_6 are '9'	
	EC43, EC43, EC44, EC45, EC46	format is $C_1C_2C_3$ $C_4C_5C_6$, characters C_1 ,C_3 , C_5 are 'A', characters C_2 ,C_4 , C_6 are '9'	A9A 9A9
	EC43, EC43, EC44, EC45, EC46	format is $C_1C_2C_3$ $C_4C_5C_6$, characters C_1 is 'Y', C_3 , C_5 are 'Z', characters C_2 ,C_4 , C_6 are '0'	Y0Z 0Z0
Postal Code (invalid)	EC47	length=5	H5H5H
	EC48	length=8	J3K 2M6L
	EC49	C_1 is 'D'	D5B3C4
	EC49	C_1 is 'U'	U5B3C4
	EC50	C_1 is 'W'	W5B3C4
	EC50	C_1 is 'Z'	Z5B3C4
	EC51	C_1 is 'a'	a6H2K2
	EC51	C_1 is 'z'	z6H2K2
	EC52	C_1 not letter	@5C4X2, [5C4X2
	EC53	C_2 not a digit	H/K7L6, H:K7L6
	EC54	C_3 is 'D'	K5D3C4
	EC54	C_3 is 'U'	K5U3C4
	EC55	C_3 is 'a'	K6a2K2
	EC55	C_3 is 'z'	H6z2K2
	EC56	C_3 not letter	L5@4X2, M5[4X2
	EC57	C_4 not a digit	H1KAL6, H2K:L6
	EC58	C_5 is 'D'	K5L3D4

Input Field	ECs	Boundary Condition	Sample Values
	EC58	C_5 is ‘U’	K5H3U4
	EC59	C_5 is ‘a’	K6X2a2
	EC59	C_5 is ‘z’	H6M2z2
	EC60	C_5 not letter	L5L4@2, M5H4[2
	EC61	C_6 not a digit	H1K1L/, H2K5L:

Assumptions:

- The ASCII table character ordering was used to find boundary values for alphabetic letters.
- Boundary values are not listed for the output as each equivalence class consists of a single value.

Test Case Derivation

The main objective is to cover each equivalence class when possible. In this case, because of interface constraints, we won’t be able to derive tests that cover classes EC40 and EC41. We are also unable to force outputs as specified for EC74.

Different strategies may be used to cover equivalence classes:

- One-to-one - we provide at least one test case per equivalence class.
- Minimized – we provide a minimum number of test cases, each covering as much ECs as possible.
- Myer’s selection approach – one-to-one strategy for invalid ECs and minimized for valid ECs.
- Combinatorial approaches – different degree of combination among the various classes (e.g. pair-wise).

We may also select to use all the boundary values identified or only as needed.

Suppose a Test Suite based on Myer’s selection approach using boundary values as needed.

Sample Test Cases for Valid Inputs

Test Case	Input	Expected Output	Equivalence Classes
1	UserName: A00000 FirstName: “” LastName: “” Age: 18	registration request accepted	EC1, EC2, EC3 EC9 EC14 EC19

Test Case	Input	Expected Output	Equivalence Classes
	Email: +@-.AA City: Halifax PostalCode: A0A0A0		EC24, EC25, EC26, EC27, EC28, EC29, EC62
2	UserName: zzzzzzzzzzzz FirstName: A LastName: zzzzzzzzzzzzzzzzz Age: 64 Email:zzZZz.zz999@zzz.zzzz99999zzzzzzz.zzzzzzz9999zzzzz.zzzzzz City:Toronto PostalCode: Y9Z9Z9	registration request accepted	EC1, EC2, EC3, EC10, EC11, EC14, EC15, EC16, EC19, EC24, EC25, EC26, EC27, EC28, EC29, EC39, EC42, EC43, EC44, EC45, EC46, EC62
3	UserName: Aure723xxz FirstName: “A ” LastName: “Bob The Great” Age: 35 Email: boByy4534@some.where.com City: Ottawa PostalCode: A9A 9A9	registration request accepted	EC1, EC2, EC3, EC10, EC11, EC15, EC16, EC19, EC39, EC43, EC43, EC44, EC45, EC46, EC62
4	UserName: X56565z0 FirstName: Bond LastName: James Age: 60 Email: jb007@mi6.org City: Montreal PostalCode: Y0Z 0Z0	registration request accepted	EC1, EC2, EC3, EC10, EC11, EC14, EC15, EC16, EC19, EC24, EC25, EC26, EC27, EC28, EC29, E41, EC43, EC43, EC44, EC45, EC46

Question 1: Provide two additional test cases for valid inputs covering Equivalence Classes not covered above.

<u>Sample Test Cases for Invalid Inputs</u>

Test Case	Input	Expected Output	Equivalence Classes
5	UserName: “” FirstName: “” LastName: “” Age: 18 Email: +@-.AA	Err1	EC4, EC63

Test Case	Input	Expected Output	Equivalence Classes
	City: Halifax PostalCode: A0A0A0		
6	UserName: z FirstName: A LastName: zzzzzzzzzzzzzzzzz Age: 64 Email:zzZZz.zz999@zzz.zzzz99999zzzzzzz.zzzzzzz9999zzzzz.zzzzzz City:Toronto PostalCode: Y9Z9Z9	Err1	EC5, EC63
7	UserName: zzzzz FirstName: “A ” LastName: “Bob The Great” Age: 35 Email: boByy4534@some.where.com City: Ottawa PostalCode: A9A 9A9	Err1	EC6, EC63
8	UserName: @adr278a FirstName: Bond LastName: James Age: 60 Email: jb007@mi6.org City: Montreal PostalCode: Y0Z 0Z0	Err1	EC7, EC63

Question 2: Provide two additional test cases for invalid inputs covering Equivalence Classes not covered above.