# **Práctica IX – Pruebas unitarias**

La meta de esta práctica es probar el correcto funcionamiento de diversas funciones. Se realizarán primero las técnicas de caja negra y caja blanca para planificar las pruebas. En un archivo adjunto, están realizadas las pruebas de caja blanca en el propio código, haciendo un Test Method para cada caso.

Las pruebas serán realizadas por Alejandro Afonso Barber, usando diferentes herramientas para hacer las técnicas de caja negra y caja blanca,y usando Visual Studio Community en una máquina virtual.

###### string[] PersonasPorEncimaMedia(Personas[] listaPersonas)

Clases de equivalencia

| **Parámetro** | **Válido** | **Inválido** |
| --- | --- | --- |
| media | n > 0 (1) | n < 0 (2) |
| personas | Personas.Lenght > 0 (3) | null (4) |

Casos Limite

| **Caso Límite** | **Entrada** | **Resultado Esperado** |
| --- | --- | --- |
| (5)LIMITE INFERIOR = 0 | media = 0, personas = [{nombre: "Ana", apellido: "García", altura: 0.00}] | ["Ana García"] |
| (6) | media = 0, personas = [{nombre: "Ana", apellido: "García", altura: 0.01}] | ["Ana García"] |
| (7)LIMITE SUPERIOR = 3 | media = 3.00, personas = [{nombre: "Ana", apellido: "García", altura: 2.99}] | - |
| (8) | media = 3.00, personas = [{nombre: "Ana", apellido: "García", altura: 3.00}] | ["Ana García"] |

CASOS DE PRUEBA

| **Parámetros** | **Output** | **Clases de equivalencia** |
| --- | --- | --- |
| media = 4.5, personas = null | Error: personas debe ser un array que no esté vacío | 1, 4, 6 |
| media = -1 personas = [{nombre: "Ana", apellido: "García", altura: 1.00}] | Error: media no puede ser menor que 0 | 2, 3 |
| media = "casa", personas = [ ] | Error: media debe ser un número | 3, 4 |
| media = 4.5, personas = [{nombre: "Ana", apellido: "García", altura: 0.00}] | [] | 1, 5 |
| media = 4.5, personas = [{nombre: "Ana", apellido: "García", altura: 0.01}] | [] | 2, 5 |
| media = 2.00, personas = [{nombre: "Ana", apellido: "García", altura: 1.99}] | [] | 3, 5 |
| media = 3.00, personas = [{nombre: "Ana", apellido: "García", altura: 3.00}] | ["Ana García"] | 1, 3, 7 |
| media = -5, personas = [{nombre: "Ana", apellido: "García", altura: 2.99}] | Error: media debe ser mayor o igual a cero y un número | 2,3,7 |

Pruebas de caja blanca:

v(f):5-4+2=3



Caminos:

1=1,2,4,Fin

2=1,3,Fin

3= 1,2.4,1… Fin

Caso de prueba:

| Input | Output | Camino |
| --- | --- | --- |
| media = 1 altura = 1 nombre =Ale | Alejandro | 1 |
| media = 2 altura = 1 nombre =Ale | [] | 2 |
| media = 2 altura = 3 nombre =Ale  altura = 1 nombre =Ale1 | Ale | 3 |

###### string[] PersonasPorDebajoMedia(Personas[] listaPersonas)

Clases de equivalencia

| **Parámetro** | **Válido** | **Inválido** |
| --- | --- | --- |
| media | n > 0 (1) | n < 0 (2) |
| personas | Personas.Lenght > 0 (3) | null (4) |

Casos Limite

| **Caso Límite** | **Entrada** | **Resultado Esperado** |
| --- | --- | --- |
| (5)LIMITE INFERIOR = 0 | media = 0, personas = [{nombre: "Ana", apellido: "García", altura: 0.00}] | - |
| (6) | media = 0, personas = [{nombre: "Ana", apellido: "García", altura: 0.01}] | - |
| (7)LIMITE SUPERIOR = 3 | media = 3.00, personas = [{nombre: "Ana", apellido: "García", altura: 2.99}] | ["Ana García"] |
| (8) | media = 3.00, personas = [{nombre: "Ana", apellido: "García", altura: 3.00}] | - |

CASOS DE PRUEBA

| **Parámetros** | **Output** | **Clases de equivalencia** |
| --- | --- | --- |
| media = 4.5, personas = null | Error: personas debe ser un array que no esté vacío | 1, 4, 6 |
| media = -1 personas = [{nombre: "Ana", apellido: "García", altura: 1.00}] | Error: media no puede ser menor que 0 | 2, 3 |
| media = "casa", personas = [ ] | Error: media debe ser un número | 3, 4 |
| media = 4.5, personas = [{nombre: "Ana", apellido: "García", altura: 0.00}] | ["Ana García"] | 1, 5 |
| media = 4.5, personas = [{nombre: "Ana", apellido: "García", altura: 0.01}] | ["Ana García"] | 2, 5 |
| media = 2.00, personas = [{nombre: "Ana", apellido: "García", altura: 1.99}] | ["Ana García"] | 3, 5 |
| media = 3.00, personas = [{nombre: "Ana", apellido: "García", altura: 3.00}] | [] | 1, 3, 7 |
| media = -5, personas = [{nombre: "Ana", apellido: "García", altura: 2.99}] | Error: media debe ser mayor o igual a cero y un número | 2,3,7 |

Pruebas de caja blanca:

v(f):5-4+2=3



Caminos:

1=1,2,4,Fin

2=1,3,Fin

3= 1,2.4,1… Fin

Caso de prueba:

| Input | Output | Camino |
| --- | --- | --- |
| media = 1 altura = 1 nombre =Ale | [] | 1 |
| media = 2 altura = 1 nombre =Ale | Ale | 2 |
| media = 2 altura = 1 nombre =Ale  altura = 3 nombre =Ale1 | Ale | 3 |

###### ***String[] MostrarPersonas(string[] personasFiltradas, string mensajeFiltro)***

Clases de equivalencia

| **Parámetro** | **Válido** | **Inválido** |
| --- | --- | --- |
| bool | true (1) | null (3) |
| bool | false (2) | null (3) |
| personas | Personas.Lenght > 0 (4) | null (5) |

Casos Limite

| **Caso** | **Descripción** | **personasFiltradas** | **listaPersonas** | **MensajeFiltroEsValido** | **Resultado Esperado** |
| --- | --- | --- | --- | --- | --- |
| (6) | No se ha filtrado nada | [] | [...] | true | [] |
| (7) | No se ha filtrado nada | [] | [...] | false | [] |
| (8) | Se ha filtrado todo | [...] | [...] | true | [...] |
| (9) | Se ha filtrado todo | [...] | [...] | false | [...] |
| (10) | Bool es null | [...] | [...] | null | Error |

CASOS DE PRUEBA

| **Parámetros** | **Output** | **Clases de equivalencia** |
| --- | --- | --- |
| bool = true  personasFiltradas = [...] | personasFinal = [...] | 1,4,8 |
| bool = false  personasFiltradas = [...] | personasFinal = [...] | 2,4,9 |
| bool = true  personasFiltradas = [] | personasFinal = [] | 1,5,6 |
| bool = false  personasFiltradas = [] | personasFinal = [] | 2,5,7 |
| bool = null  personasFiltradas = [...] | personasFinal = [] | 3,4,10 |
| bool = null  personasFiltradas = [] | personasFinal = [] | 3,5,10 |

Pruebas de caja blanca:



v(f):4-4+2=2

| Input | Output | Camino |
| --- | --- | --- |
| bool = true | PersonasFinal | 1,2,F |
| bool = false | PersonasFinal | 1,3,F |

###### String personaMasAlta(Personas[] listaPersonas)

Clases de equivalencia

| **Parámetro** | **Válido** | **Inválido** |
| --- | --- | --- |
| Personas[] listaPersonas | Count > 0 (1) | Count < 0 / null (2) |
| Personas[] listaPersonas | Count = 0 (3) | Count < 0 / null (2) |

Casos Limite

| **Caso Límite** | **Entrada** | **Resultado Esperado** |
| --- | --- | --- |
| Count  LIMITE INFERIOR = 0 | lIstaPersonas.Count = 0 (5) | * (6) |
|  | lIstaPersonas.Count = -1 (7) | Error (8) |
| Limite superior > 0 | listaPersonas.Count > 0 (9) | String Persona mas alta (10) |

CASOS DE PRUEBA

| **Parámetros** | **Output** | **Clases de equivalencia** |
| --- | --- | --- |
| lIstaPersonas.Count = 0 | String vacio | 3,5,6 |
| lIstaPersonas.Count = -1 | Error | 2,7,8 |
| lIstaPersonas.Count > 1 | String PersonaMasAlta | 1,9,10 |

PRUEBAS CAJA BLANCA:



v(f):8-6+2=4

| Input | Output | Camino |
| --- | --- | --- |
| listaPersonas = 0 | - | 1,2,F |
| listaPersonas = 1 | PersonaFinal | 1,3,5,F |
| listaPersonas = 2 | PersonaFinal | 1,3,4,3,5,F |
| listaPersonas = 2 | PersonaFinal | 1,3,5,3,4,F |

###### ***Public static String strongestPokemon(string rutaFichero, int generation)***

Clases de equivalencia

| **Parámetro** | **Válido** | **Inválido** |
| --- | --- | --- |
| string fichero | Exist (1) | null (2) |
| generation | generation > 1 & generation < 7 (3) | generation < 0 & generation > 6 (4) |

Casos Limite

| **Caso Límite** | **Entrada** | **Resultado Esperado** |
| --- | --- | --- |
| generation  LIMITE INFERIOR = -1 | Generation = -1 (5) | * Error (6) |
| LIMITE INFERIOR = 0 | Generation = 0 (7) | Error (6) |
| LIMITE INFERIOR = 1 | Generation = 1 (9) | PokemonName (10) |
| LIMITE SUPERIOR = 6 | Generation = 6 (11) | PokemonName (10) |
| Limite superior > 6 | generation = 7 (12) | Error (6) |

CASOS DE PRUEBA

| **Parámetros** | **Output** | **Clases de equivalencia** |
| --- | --- | --- |
| generation = x  fichero = false | Error | 2,4 |
| fichero = true  generation = 0 | Error, generacion no valida | 1,4,6 |
| fichero = true  generation = 1 | PokemonName | 1,3,9,10 |
| fichero = true  generation = 6 | PokemonName | 1,3,10,11 |
| fichero = true  generation = 8 | Generation no valida | 1,4,6,12 |

Pruebas caja blanca:



v(f):15-13+2=4

| Input | Output | Camino |
| --- | --- | --- |
| file = false | Error | 1,2,F |
| file = true / | Error | 1,3,4,F |
| file = true generation = 1 | PinsirMega Pinsir | 1,3,5,6,7,8,F |
| file = true generation = 8 | “” | 13,5,9,F |
|  |  |  |

###### Public void filterPokemon(string rutaFichero)

Clases de equivalencia

| **Parámetro** | **Válido** | **Inválido** |
| --- | --- | --- |
| string fichero | Exist (1) | null (2) |

Casos Limite

| **Caso Límite** | **Entrada** | **Resultado Esperado** |
| --- | --- | --- |
| fichero = true | fichero (3) | * X pokemons (4) |
| fichero = false | / (5) | Error, fichero no valido (6) |

CASOS DE PRUEBA

| **Parámetros** | **Output** | **Clases de equivalencia** |
| --- | --- | --- |
| fichero = false | Error | 1,3,4 |
| fichero = true | X pokemons | 1,4,6 |

Pruebas caja blanca:



v(f):4-4+2=2

| Input | Output | Camino |
| --- | --- | --- |
| file = false | Error | 1,2,F |
| file = true | X pokemons | 1,3,F |