## NavajaSuiza.Aplicación\_1\_NumerosPrimos

### **PRUEBA DE CAJA NEGRA**

Fecha de la prueba: 09/03/2017

Persona encargada de la prueba: Antonio Victoriano Pérez-Castejón Martínez

### - Clases de equivalencia

- o **iNumero**: ha de ser un número entero mayor que 0
  - iNumero = = número entero positivo > 0.
     Valores límite: 1, 2, 2.147.483.646, 2.147.483.647, 2.147.483.648
  - iNumero == 0
  - iNumero = = número entero negativo.
     Valores límite: -2.147.483.649, -2.147.483.648, -2.147.483.647, -2, -1.
  - iNumero = = número real.
  - iNumero = = cadena de caracteres alfanuméricos
  - iNumero = = textbox vacío

## - Casos de prueba

Salida → Valor = resultado esperado o correcto

Salida  $\rightarrow$  Valor = resultado no esperado o incorrecto (Valor) = resultado esperado

CLASES EQUIVALENCIA	CASOS DE PRUEBA	SALIDA
	iNumero = 1	Primo
	iNumero = 2	Primo
iNumero == número entero positivo > 0	iNumero = 3	Primo
Valores límite: 1, 2, 2.147.483.646,	iNumero = 4	No primo
2.147.483.647, 2.147.483.648	iNumero = 2.147.483.646	No primo
	iNumero = 2.147.483.647	Primo
	iNumero = 2.147.483.648	No es un número entero correcto
iNumero == 0	iNumero = 0	Primo (Error: el 0, por definición, no se considera un número primo)
	iNumero = -2.147.483.649	No es un número entero correcto
	iNumero = -2.147.483.648	Primo (Error: el número
iNumero = = número entero		introducido es negativo)
negativo. Valores límite:	iNumero = -2.147.483.647	Primo
-2.147.483.649,		(Error: el número introducido es negativo)
-2.147.483.648,	iNumero = -2	Primo
-2.147.483.647, <b>-2, -1</b> .		(Error: el número
		introducido es negativo)
	iNumero = -1	Primo
		(Error: el número
		introducido es negativo)
201	'N 2 7	No es un número entero
iNumero = = número real	iNumero = 2,7	correcto
iNumero = = cadena de caracteres alfanuméricos	iNumero = 5abcd	No es un número entero correcto
iNumero = = textbox vacío	iNumero = ""	No es un número entero correcto

### **PRUEBA DE CAJA BLANCA**

Fecha de la prueba: 09/03/2017

bool ComprobarPrimo(int iNumero)

Persona encargada de la prueba: Antonio Victoriano Pérez-Castejón Martínez

```
bool bEsPrimo = true;
int iModulo = 0;
int i = 2;

while(i <= iNumero / 2 && bEsPrimo)
{
    iModulo = iNumero % i;
    if(iModulo == 0)
    {
        bEsPrimo = false;
    }
    i++;
}

return bEsPrimo;
}

Salida → Valor = resultado esperado o correcto
Salida → Valor = resultado no esperado o incorrecto (Valor) = resultado esperado</pre>
```

COBERTURA	CASOS	SALIDA
	<inumero></inumero>	
(1)	<1>	Primo
Entrar en el While 0 veces	<2>	Primo
	<3>	Primo
(1)	<4>	No primo
Entrar en el While 1 vez	<5>	Primo
	<6>	No primo
(1) Entrar en el While varias	<7>	Primo
veces	<35>	No primo
(2)	<4>	No primo
Entrar en el If: iModulo == 0	<6>	No primo
	<66>	No primo
(2)	<3>	Primo
No entrar en el If: iModulo != 0	<5>	Primo
	<11>	Primo

## Tabla simplificada

COBERTURA	CASOS <inumero></inumero>	SALIDA
(1) Entrar en el While 0 veces	<1>, <2>, <3>	Primo
(1)	<4>, <6>	No primo
Entrar en el While 1 vez	<5>	Primo
(1) Entrar en el While varias	<7>	Primo
veces	<35>	No primo
(2) Entrar en el If: iModulo == 0	<4>, <6>, <66>	No primo
(2) No entrar en el If: iModulo != 0	<3>, <5>, <11>	Primo

# NavajaSuiza. Aplicación\_4\_Potencia Numero

### **PRUEBA DE CAJA NEGRA**

Fecha de la prueba: 09/03/2017

Persona encargada de la prueba: Antonio Victoriano Pérez-Castejón Martínez

### - Clases de equivalencia

- o **iBase**: ha de ser un número entero (positivo o negativo)
  - iBase == número entero positivo.

Valores límite: 0, 1, 2, 2.147.483.646, 2.147.483.647, 2.147.483.648

- iBase == número entero negativo.
  - Valores límite: -2.147.483.649, -2.147.483.648, -2.147.483.647, -2, -1.
- iBase == número real.
- iBase == cadena de caracteres alfanuméricos
- iBase == ""
- o **iExponente**: ha de ser un número entero (positivo o negativo)

iExponente == número entero positivo.

Valores límite: 0, 1, 2, 2.147.483.646, 2.147.483.647, 2.147.483.648

iExponente == número entero negativo.

Valores límite: -2.147.483.649, -2.147.483.648, -2.147.483.647, -2, -1.

- iExponente == número real.
- iExponente == cadena de caracteres alfanuméricos
- iExponente == ""

## - Casos de prueba

Salida → Valor = resultado esperado o correcto

Salida  $\rightarrow$  Valor = resultado no esperado o incorrecto (Valor) = resultado esperado

CLASES EQUIVALENCIA	CASOS DE PRUEBA	SALIDA
	iBase = 0	1
	iExponente = 0	
	iBase = 0	0
	iExponente = 1	
	iBase = 0	0
	iExponente = 2	
	iBase = 0	0
	iExponente = 3	
	iBase = 0	0
	iExponente = 4	
iBase == número entero	ina.	
positivo. Valores límite 0, 1, 2,	iBase = 0	0
2.147.483.646, 2.147.483.647,	iExponente = 2.147.483.646	0
2.147.483.648		
iEvnananta número entero	iBase = 0	Error
<b>iExponente</b> == número entero positivo. Valores límite 0, 1, 2,	iExponente = 2.147.483.647	2.70.
2.147.483.646, 2.147.483.647,	·	El programa se congela
2.147.483.648	iBase = 0	¡Error!
	iExponente = 2.147.483.648	
		El exponente introducido
		debe de ser un número
		entero
	iBase = 1	1
	iExponente = 0	
	iBase = 1	1
	iExponente = 1	1
	ityponente – 1	
	iBase = 1	1
	iExponente = 2	
	iBase = 1	1
	iExponente = 3	
	iBase = 1	1
	iExponente = 4	

	iBase = 1	1
		1
	iExponente = 2.147.483.646	
	iBase = 1	Error
	iExponente = 2.147.483.647	Elloi
	1Exponence = 2.147.403.047	El programa se congela
	iBase = 1	¡Error!
	iExponente = 2.147.483.648	1211011
	12xponente 2.117.103.010	El exponente introducido
		debe de ser un número
		entero
	iBase = 2	1
	iExponente = 0	_
	1EXPONENCE 0	
	iBase = 2	2
	iExponente = 1	
	iBase = 2	4
	iExponente = 2	
	iBase = 2	8
	iExponente = 3	
	iExponence – 3	
	iBase = 2	16
	iExponente = 4	
	·	
	iBase = 2	0
	iExponente = 2.147.483.646	
	iBase = 2	Error
	iExponente = 2.147.483.647	
		El programa se congela
	iBase = 2	¡Error!
	iExponente = 2.147.483.648	
		El exponente introducido
		debe de ser un número
		entero
	iBase = 2.147.483.646	1
	iExponente = 0	
	:Dage 2147 402 C4C	2147402646
	iBase = 2.147.483.646	2147483646
	iExponente = 1	
	iBase = 2.147.483.646	4
	iExponente = 2	·
	iExponence - 2	
	iBase = 2.147.483.647	1
	iExponente = 0	_
	ponence	
I .	1	ı

		24.47.400.617
	iBase = 2.147.483.647	2147483647
	iExponente = 1	
	iBase = 2.147.483.647	1
	iExponente = 2	
	iBase = 2.147.483.648	¡Error!
	iExponente = 0	to have to too deaths do had
		La base introducida debe
		de ser un número entero
	iBase = 2.147.483.648	¡Error!
	iExponente = 1	
		La base introducida debe
		de ser un número entero
	iBase = 2.147.483.648	¡Error!
	iExponente = 2	
		La base introducida debe
· · · · · · · · · · · · · · · · · · ·		de ser un número entero
iBase == número entero	iBase = -1	¡Error!
negativo. Valores límite:	iExponente =	
-2.147.483.649,	-2.147.483.649	El exponente introducido
-2.147.483.648,		debe de ser un número
-2.147.483.647,		entero
-2, -1.	iBase = -1	Error
	iExponente =	
	-2.147.483.648	El programa se congela
iExponente == número entero	iBase = -1	Error
negativo.	iExponente =	
Valores límite:	-2.147.483.647	El programa se congela
-2.147.483.649,	iBase = -1	-1
-2.147.483.648,	iExponente = -1	
-2.147.483.647,		
-2, -1.	iBase = -1	1
	iExponente = -2	
	·D	-1
	iBase = -1	
	iExponente = -3	
	iBase = -1	1
		1
	iExponente = -4	
	iBase = -2	0
	iExponente = -1	(-0,5)
	iBase = -2	0
	iExponente = -2	(-0,25)
iBase == número entero	iBase = 0	Error de excepción
positivo. Valores límite = 0, 1, 2	iExponente = -1	
	iBase = 0	Error de excepción
	L	

iExponente == número negativo. Valores límite -1, -2	iExponente = -2	
negativo. valores illilite -1, -2	iBase = 1	1
	iExponente = -1	
	iBase = 1	1
	iExponente = -2	
	iBase = 2	0
	iExponente = -1	(0,5)
	iBase = 2	0
	iExponente = -2	(0,25)
iBase == número entero	iBase = -1	1
negativo. Valores límite = -1, -2	iExponente = 0	
iExponente == número positivo.	iBase = -1	-1
Valores límite = 0, 1 y 2	iExponente = 1	
	iBase = -1	1
	iExponente = 2	
	iBase = -2	1
	iExponente = 0	
	iBase = -2	-2
	iExponente = 1	
	iBase = -2	4
	iExponente = 2	
iBase == número real	iBase = 2,5	¡Error!
iExponente == número real	iExponente == 3,5	La base introducida debe
rexponente numero real	iexponente 3,5	de ser un número entero
		El exponente introducido
		debe de ser un número
		entero
iBase == número real		¡Error!
iExponente == número entero	iBase = 3,7	[LITOI:
		La base introducida debe
	iExponente == -2	de ser un número entero

	1	1
iBase == número entero	iBase = 5	:Errorl
ibase == numero entero	IBase = 5	¡Error!
iExponente == número real	iExponente = 3,8	El exponente introducido
		debe de ser un número
		entero
iBase == cadena de caracteres	iBase = abcd	¡Error!
alfanuméricos		
:Former and a section of	iExponente = a5	La base introducida debe de ser un número entero
iExponente == cadena de caracteres alfanuméricos		de ser un numero entero
		El exponente introducido
		debe de ser un número
		entero
iBase == cadena de caracteres	iBase = 55ac	¡Error!
alfanuméricos		
iExponente == número entero	iExponente = 10	La base introducida debe de ser un número entero
lexponente numero entero		de ser dir numero entero
iBase == número entero	iBase = 6	¡Error!
iExponente == cadena de	iExponente = xyz	El exponente introducido
caracteres alfanuméricos	IEXPONENCE - XyZ	debe de ser un número
		entero
		¡Error!
iBase == textbox vacío	iBase == ""	[EITOI:
		La base introducida debe
iExponente == textbox vacío	iExponente == ""	de ser un número entero
		El exponente introducido
		debe de ser un número
		entero
iBase == textbox vacío	iBase == ""	¡Error!
iExponente == número entero	iExponente == 15	La base introducida debe
		de ser un número entero
iBase == número entero	iBase == 8	¡Error!
iExponente == textbox vacío	iExponente == ""	El exponente introducido
,	,	debe de ser un número
		entero

### **PRUEBA DE CAJA BLANCA**

Fecha de la prueba: 09/03/2017

Persona encargada de la prueba: Antonio Victoriano Pérez-Castejón Martínez

```
int CalcularPotencia(int iBase, int iExponente)
{
   int iResultado = 1;
   bool bPositivo = true;

   if (iExponente < 0)
   {
      bPositivo = false;
      iExponente = -iExponente;
   }

   for (int i = 1; i <= iExponente; i++)
   {
      iResultado = iResultado * iBase;
   }

   if (bPositivo == false)
   {
      iResultado = 1 / iResultado;
   }

   return iResultado;
}</pre>
```

Salida → Valor = resultado esperado o correcto
Salida → Valor = resultado no esperado o incorrecto (Valor) = resultado esperado

COBERTURA	CASOS <ibase, iexponente=""></ibase,>	SALIDA
(1)	<6, -5>	0
Entrar en el primer If:		
iExponente < 0		(0,0001286008230452675)
(4)		
(1)	<6, 0>	1
No entrar en el primer If:		
iExponente >= 0	<6, 6>	46656
(2)	<8, 0>	1
Entrar en el for 0 veces	10,07	1
(2)	<9, 1>	9
Entrar en el for 1 vez	(3, 12	3
(2)	<10, 3>	1000
Entrar en el for 1 varias	,	
veces		
(3)	<4, -3>	0
Entrar en el segundo If:		(0,015625)
bPositivo == false		
(3)	<6, 8>	1679616
No entrar en el segundo If:		
bPositivo == true		