Evaluación 2

1. Considere las funciones eje1 y eje2 definidas en el archivo 2Maquina. Evalúe eje1({1}) y eje2(2). ¿Qué resultado se obtiene? Explique por qué a pesar de no tener el dominio incluido en las definiciones, el resultado no es indefinido.

```
 >     eje1:=func(x);
>>     if (x>2) then return 3 else return 1;
>>     end;
>>     end;
>     eje1({1});

*** if (x>2) then return 3 else return 1;

!Error: Bad arguments in:
2 <relation> {1};


>     eje2:=func(x,y);
>>     local a;
>>     a:= x and y;
>>     if a then return (x or y) else return not (x or y);
>>     end;end;
>     eje2(2);

!Error: Too few arguments
>
```

Según lo que yo obtuve, el resultado no es indefinido pero el programa no sabe qué resultado dar porque yo le pedí que evaluara las funciones en elementos que no son parte del dominio. En el primer caso, el dominio es el conjunto de los números reales, y yo le pedí que lo evaluara en un conjunto **¿en cuál??,** por eso no sabe qué devolver. Si bien no se definió el dominio, sí se especificó que antes de evaluar se tiene que discernir si el elemento a evaluar verifica x>2 o no. Y eso no se puede hacer con el conjunto {1},**ya que el programa no tiene definida una relación de orden entre un conjunto y un número.** Entonces no sabe qué hacer, y por eso identifica que el argumento que se dio no es correcto.

ISetL toma como dominio los números reales, la razón es la que marqué en verde en tu texto.

En el segundo caso el dominio es el producto cartesiano de dos conjuntos, y yo le pedí que lo evaluara en un número, por eso no sabe qué hacer. La máquina espera como valor a evaluar un par ordenado, y cuando se le da sólo un elemento, sea lo que sea, ya se da cuenta que le faltan argumentos y eso es lo que responde. Aquí tampoco se explicita el dominio pero el programa tiene las herramientas suficientes para darse cuenta que lo que se da para evaluar es incorrecto.

2. Describa los posibles resultados de la aplicación de una función, considerando el indefinido como uno de ellos

No entiendo bien la pregunta. Supongo que el posible resultado de la aplicación de una función depende del codominio: puede ser un número, un punto, una variable booleana, un nombre, un par ordenado, un conjunto, una función, etc. Si el elemento que le doy para aplicar no está en el dominio, entonces la respuesta será indefinida.

¿y el error? El resultado puede ser un elemento del co-dominio, el indefinido o error.

3. ¿Puede el dominio de una función ser cualquier conjunto? ¿Y el co dominio? Justifique.

En mi opinión sí, siempre y cuando se cumpla que el conjunto imagen del dominio esté incuido en el codominio.

4. ¿Cuáles son el dominio y el co-dominio de las funciones "derivada" e "integral" usadas en el cálculo matemático?

El dominio es el conjunto de las funciones derivables o integrables, respectivamente. Y el codominio es el conjunto de todas las funciones. Vale aclarar que el concepto de función derivable o integrable puede variar según el contexto en el que se está trabajando y la definición que se adopta en el mismo.

5. Defina en ISetL una función is_natural tal que devuelva true si el argumento es un número natural. *Pista: use la función is_integer.*

```
 is_natural:=func(x);
>>    if (is_integer(x) and x>-1) then return true else return false;
>>    end;
>>    end;

Pruebas:
>     is_natural(9);
true;
>     is_natural(2.5);
false;
>     is_natural(0);
true;
>     is_natural({1});
false;
```

6. Implemente en ISetL cada una de las funciones especificadas en los ejercicios del 1 al 6 de la Actividad 2, usando la función is_natural definida anteriormente.

Ejercicio 1:

```
>     divisores:=func(x);
>>    return {y : y in {1..x} | (is_natural(y) and x mod y = 0)};
>>    end;

Pruebas:
>     divisores(39);
{13, 39, 3, 1};
>     divisores(4.5);

*** return {y : y in {1..x} | (is_natural(y) and x mod y = 0)};

!Error: Bad args in low..high
>     divisores(360);
{60, 45, 72, 90, 120, 180, 360, 36, 40, 24, 30, 5, 6, 1, 2, 3, 4, 15,
18, 20, 12, 10, 9, 8};
```

Ejercicio 2:

```
>     es_primo:=func(x);
>>    if divisores(x)={1,x} then return true else return false;
>>    end;
>>    end;

Pruebas:
>     es_primo(26);
false;
>     es_primo(321);
false;
>     es_primo(47);
true;
```

Podrías escribir return divisores(x)={1,x}
Esa expresión ya tiene un valor verdadero o falso.

Ejercicio 3:

```
>     dos_en:=func(x);
>>    if is_set(x) then return {y : y in pow(x) | 2 in y};
>>    end;
>>    end;

Pruebas:
>     dos_en({4,2,7});

{{7, 2}, {7, 4, 2}, {2}, {2, 4}};

>     dos_en({-1,2,0});
{{2, 0}, {2, 0, -1}, {2}, {-1, 2}};
{{2}, {-1, 2}, {2, 0}, {-1, 0, 2}};
>     dos_en({0,1});

{};
```

Aún no me doy cuenta cómo hacer que x sólo sea un conjunto de naturales, por ahora quedó definido para cualquier conjunto.

Cambiar la condicón del if por esta:

```
if (is_set(x) and (forall y in x | is_nat(y))) …
```

Ejercicio 4:

```
>     equis_en:=func(x,y);
>>    return {z : z in pow(y) | x in z};
>>    end;

Pruebas:
>     equis_en(3,{1, 3, 5});
{{5, 1, 3}, {5, 3}, {3, 1}, {3}};
>     equis_en(0, {1, 3, 5});
{};
```

Ejercicio 5:

```
>     multiplos:=func(x);
>>    return ({y : y in {x..325} | (is_natural(y) and y mod x = 0)}
union {0});
>>    end;

Pruebas:
>     multiplos(7);
{105, 112, 119, 147, 140, 126, 133, 189, 182, 196, 203, 175, 168, 154,
161, 63, 70, 56, 49, 77, 84,
 91, 98, 42, 35, 28, 21, 14, 0, 7, 210, 217, 224, 238, 245, 231, 315,
322, 308, 287, 301, 294, 266,
 273, 280, 259, 252};
>     multiplos(2.5);

*** return ({y : y in {x..325} | (is_natural(y) and y mod x = 0)}
union {0});

!Error: Bad args in low..high
>     multiplos(47);
{282, 235, 141, 188, 0, 47, 94};
```

Ejercicio 6:

```
>     mul_con_tope:= func(x,y);
>>    return {z : z in multiplos(x) | z < y};
>>    end;

Pruebas:
>     mul_con_tope(3,19);
{6, 9, 3, 0, 18, 15, 12};
>     mul_con_tope(5,89);
{35, 40, 30, 60, 55, 50, 45, 65, 70, 85, 80, 75, 25, 20, 10, 15, 0,
5};
>
```