1 Define las funciones sind y cosd que calculen seno y coseno en grados.

```
sind(x):=((sin(x) · 180)/%pi);
```

$$\text{sind}(x):=\frac{\sin(x)\ 180}{\pi}$$

```
cosd(x):=((cos(x) · 180)/%pi);
```

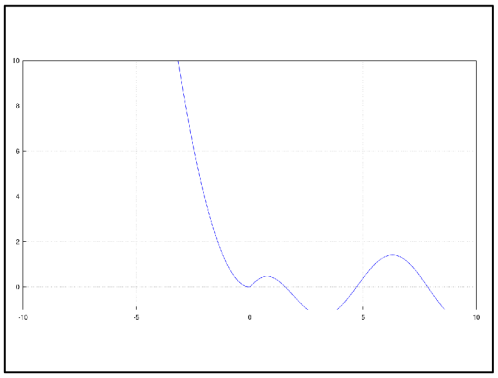
$$\text{cosd}(x):=\frac{\cos(x)\ 180}{\pi}$$

2 Define una función a trozos que valga x^2 en R−; cos(x)*arctan(x) en [0,+∞[

```
f(x):= if x<0 then x^2 else (cos(x) · atan(x));
```

$$\text{f}(x):=\text{if}\ \ x<0\ \ \text{then}\ \ x^2\ \ \text{else}\ \ \cos(x)\ \text{atan}(x)$$

```
wxdraw2d(explicit(f(x), x, −10, 10),
  yrange=[−1,10],
  grid=true,
  yaxis=true,
  xaxis=true,
  proportional_axes=xy
);
```



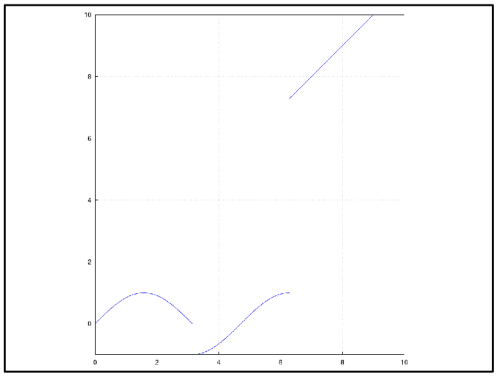
3 Define una función a trozos que valga sin(x) en [0 y π[; cos(x) en [π,2π[; x+1 en [2π,10[.

Se ha de tener en cuenta que las líneas verticales ubicadas en los saltos de la función a trozos no son reales y no deben tenerse en cuenta: son un error de graficado propio de maxima. Si quisiéramos graficar correctamente esta función a trozos, habría que definir una función distinta para cada tramo y luego graficarlas juntas. Pero eso no es lo que pide el ejercicio.

```
g(x):= if 0<=x and x<%pi then sin(x) else if %pi<=x and x<2·%pi then cos(x) else if 2·%
```

$$g(x):=\text{if } 0\le x \wedge x<\pi \text{ then } \sin(x) \text{ else if } \pi\le x \wedge x<2\,\pi \text{ then } \cos(x) \text{ else if } 2\,\pi\le x \wedge x<10 \text{ then } x+1$$

```
wxdraw2d(explicit(g(x), x, 0, 10),
   yrange=[-1,10],
   grid=true,
   yaxis=true,
   xaxis=true,
   proportional_axes=xy
);
```



4 Define una función que calcule la suma de los primeros n números impares.

La función consiste en iterar n veces y, usando la fórmula para calcular impares 2*n-1, ir acumulando la suma en la variable que previamente inicializamos a 0.

```
sumaimpares(n):=block(
   suma: 0,
   for i:1 thru n do (impar_i: 2·i-1 , suma: suma+impar_i),
   suma
);
```

$$\text{sumaimpares}(n):=\text{block}(suma:0,\text{for } i \text{ thru } n \text{ do } (impar\_i:2\,i-1,suma:suma+impar\_i),suma)$$

```
sumaimpares(3);
```

9

5 Define una función que calcule la suma de los múltiplos de 3 menores que 1000.

Esta función no necesita parámetro (porque nos indican específicamente los menores que 1000). Iteramos sumando sobre la variable inicializada a 0 y el programa parará en el momento que este sea mayor que 1000.

```
multiplos3():=block(
   suma: 0,
   i:3,
   while i<1000 do (suma: suma+i, i: i+3),
   suma
);
```

```
multiplos3(?):=block(suma:0,i:3,while i<1000 do
(suma:suma+i,i:i+3),suma)
```

```
multiplos3();
```

```
166833
```

6 El grosor de una hoja de papel A4 de 80 gr estándar es de 0.11 mm. ¿Cuál es su grosor después de doblarlo 5 veces? ¿Cuántas veces hay que doblarla para que su altura sea 1cm? ¿Y para que la altura sea 1.80 m? ¿Y el radio de la tierra?

Cada vez que doblamos la hoja de papel, es como si sumáramos 2^i folios: primer pliege, 2 folios. Segundo pliegue, 4 folios. Tercer pliegue, 8 folios... Acumulamos esta suma, en el primer apartado 5 veces:

```
block(
   suma: 0,
   for i:0 thru 5 do (suma: suma + 0.00011 · 2^i),
   suma
);
```

```
0.00693
```

El grosor tras doblarlo 5 veces es de 0.00694 metros, o 6.94 milímetros.

Para los siguientes apartados definimos una función que itera hasta que alcanzamos el número que introducimos. Como prerrequisito, el parámetro que le damos debe estar dado en metros. Al ser un bucle while, cabe la posibilidad de que si no comprobamos la condición (grosor actual menor que grosor pedido) iteremos una vez de más, por eso metemos el incremento del iterador en un if.

```
cuantospliegues(grosor):=block(
   a4: 0.00011, /*usamos SI, el parámetro grosor debe ser dado en METROS*/
   grosor_actual: a4,
   pliegues: 0,
   while grosor_actual < grosor do (
      grosor_actual: grosor_actual + a4 · 2^pliegues,
      if (grosor_actual < grosor) then pliegues: pliegues + 1 /*Ponemos if para asegurarn
   ),
   pliegues
);
```

cuantospliegues(*grosor*):=block(*a4*:1.1 10^{-4}, *grosor_actual*:*a4*, *pliegues*:0, while *grosor_actual*<*grosor* do (*grosor_actual*:*grosor_actual*+*a4* $2^{pliegues}$, if *grosor_actual*<*grosor* then *pliegues*:*pliegues*+1), *pliegues*)

```
cuantospliegues(0.01) /*para que la altura sea 1cm, es decir, 0.01 metros)*/;
```

6

```
cuantospliegues(1.8) /*para que la altura sea 1.80 metros*/;
```

13

```
cuantospliegues(6378000) /*para que la altura sea el radio ecuatorial de la tierra: 6378
```

35

7 Define una función que calcule el producto de los primeros quince términos de la sucesión 1, 1.1, 1.01, 1.001,...

Primero definimos una lista vacía con makelist de 15 elementos. Asignamos el valor inicial 1, y luego le añadimos el resto de términos con el bucle for. Los términos se calculan sumando 1 a 1/10^i.

```
sucesion():=block(
   lista: makelist([],i,1,15),
   lista[1]: 1,
   for i:1 thru 15 do( lista[i]: [1+(1/10^i)] ),
   lista
);
```

sucesion(?):=block(*lista*:makelist([ ], *i*, 1, 15), $lista_1$:1, for *i* thru 15 do $lista_i$:$\left[1+\frac{1}{10^{i}}\right]$, *lista*)

```
sucesion();
```

[[1.1],[1.01],[1.001],[1.0001],[1.00001],[1.000001],

[1.0000001],[1.00000001],[1.000000001],[1.0000000001],
[1.00000000001],[1.000000000001],[1.0000000000001],
[1.00000000000001],[1.000000000000001]]

8 Define una función que calcule el producto de los números (1+1/n)^n, para n=1,...,20.

```
block(
   pdto: 0,
   for i:1 thru 20 do (pdto: (1 + 1/i)^i, print("numero: ", i, ", resultado:", pdto))
);
```

numero: 1 *, resultado:* 2

numero: 2 *, resultado:* $\frac{9}{4}$

numero: 3 *, resultado:* $\frac{64}{27}$

numero: 4 *, resultado:* $\frac{625}{256}$

numero: 5 *, resultado:* $\frac{7776}{3125}$

numero: 6 *, resultado:* $\frac{117649}{46656}$

numero: 7 *, resultado:* $\frac{2097152}{823543}$

numero: 8 *, resultado:* $\frac{43046721}{16777216}$

numero: 9 *, resultado:* $\frac{1000000000}{387420489}$

numero: 10 *, resultado:* $\frac{25937424601}{10000000000}$

numero: 11 *, resultado:* $\frac{743008370688}{285311670611}$

numero: 12 *, resultado:* $\frac{23298085122481}{8916100448256}$

numero: 13 *, resultado:* $\frac{793714773254144}{302875106592253}$

numero: 14 *, resultado:* $\frac{29192926025390625}{11112006825558016}$

numero: 15 *, resultado:* $\frac{1152921504606846976}{437893890380859375}$

numero: 16 *, resultado:* $\frac{48661191875666868481}{18446744073709551616}$

numero: 17 *, resultado:* $\frac{2185911559738696531968}{827240261886336764177}$

numero: 18 *, resultado:* $\frac{104127350297911241532841}{39346408075296537575424}$

numero: 19 *, resultado:* $\frac{52428800000000000000000}{19784196556603135891239 79}$

numero: 20 *, resultado:* $\frac{27821842944695154863719 6401}{10485760000000000000000000}$

done

9 Calcula la suma de los números naturales entre 10 y 1000 que no son divisibles por 7.

Usamos las funciones mod (nos da el resto de dos números) y notequal(true/false de si dos números son distintos). Iterar entre los números indicados y, si el número da 0 al dividirlo por 7 (es decir, es múltiplo de 7) no se suma.

```
block(
   suma: 0,
   for i:10 thru 1000 do (if notequal(mod(i,7),0) then suma:suma+i),
   suma
);
```

429391

10 Calcula la media aritmética, la media geométrica y la media armónica de los primeros 100 números naturales.

Calculamos en el bucle for los valores acumulados de cada una de las fórmulas (consultar). Una vez los tenemos, aplicamos sus respectivas fórmulas. En la media geométrica se necesita una raiz de base n (100), así que lo sustituímos por su forma en exponente.

```
block(
   suma:0,
   pdto:1,
   suma_inv:0,
   numer:true,
   for i:1 thru 100 do(suma:suma+i, pdto:pdto·i, suma_inv:suma_inv+(1/i)),
   aritmetica: suma/100,
   geometrica: pdto ^ (1/100),
   armonica: 100/suma_inv,
   print("media aritmética: ", aritmetica),
   print("media geométrica: ", geometrica),
   print("media armónica: ", armonica)
)$
```

media aritmética: 50.5
media geométrica: 37.99268934483429
media armónica: 19.277563597396

11 Calcula el primer natural que cumple que: 1+1/2+...+1/n > 7.1

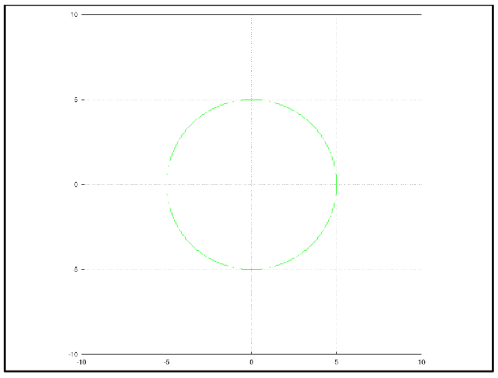
```
block(
   suma: 0,
   for i:1 while suma < 7.1 do( suma: suma + (1/i), valor: i),
   valor
);
```

680

12 Encuentra los puntos con coordenadas enteras dentro del círculo centrado en el origen y radio 5.

Para hacernos una mejor idea, primero vamos a dibujar el círculo:

```
wxdraw2d(
            color=green,
            implicit(x^2+y^2=5^2, x, −10, 10, y, −10, 10),
            color=blue,
            yrange=[−10,10],
            yaxis=true,
            xaxis=true,
            grid=true,
            proportional_axes=xy
);
```



En una circunferencia radio 5, para cada punto [a,b] perteneciente a esta se cumple que |a|,|b| <= 5, es decir, sus valores varían de -5 a 5.
Para determinar si un punto [a,b] pertenece a una circunferencia se prueba que la distancia entre dicho punto y el origen [0,0] existe una distancia de 5, el radio. Lo que debemos comprobar es entonces: $\text{sqrt}((0-a^2)+(0-b)^2) = 5$. Quitamos la raíz cuadrada, y comprobamos en 5^2 para facilitar las cosas (es decir, $(-a)^2 + (-b)^2 = 5^2$).
Así que vamos a definir dos bucles for (uno para a, otro para b) y comprobar todas las combinaciones de [a,b] entre -5 y 5.

Para almacenar los puntos hemos definido una lista de tamaño arbitrario (con un número más grande del que creíamos necesitar). El parámetro ratprint: false es para evitar que maxima imprima una traza de los cálculos.

```
block(
   lista: makelist([],i,1, 50),
   contador: 1,
   ratprint: false,
   for a:-5 thru 5 do(
      for b:-5 thru 5 do(
         if equal(((-a)^2)+((-b)^2), 5^2) then(
            lista[contador]: [a,b],
            contador: contador + 1
         )
      )
   ),
   lista
);
```

```
[[-5,0],[-4,-3],[-4,3],[-3,-4],[-3,4],[0,-5],
[0,5],[3,-4],[3,4],[4,-3],[4,3],[5,0],[],[],[],[],[],[],[],[],
[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],
[],[],[],[],[],[],[],[]]
```