Lea atentamente estas instrucciones antes de comenzar:

- Responda a cada una de las cuestiones prácticas.
- Es necesario aprobar este examen (5 puntos) para aprobar la asignatura.
- No es suficiente este examen para aprobar la asignatura: se deben aprobar los dos trabajos de la parte práctica.

Sistemas de álgebra computacional: Maxima

1. (1 punto) Abrimos una sesión de trabajo en Maxima y tecleamos

```
a:i^2$
b:1+a;
```

Indique el output que se obtiene por pantalla.

Solución: Dado que estamos usando el símbolo "i", que no tiene valor asignado, el resultado de la primera línea será simplemente asignar a la variable "a" en valor i^2 y no se mostrará por pantalla. Tras la segunda línea, que sí se mostrará por pantalla, la variabla "b" contendrá la expresión $1 + i^2$.

2. (1 punto) Abrimos una sesión de trabajo en Maxima y tecleamos

```
a:trigsimp(cos(x)^2 + sin(x)^2)$
b:sqrt(-a);
```

Indique el output que se obtiene por pantalla.

Solución: Maxima conoce la identidad trigonométrica $\cos^2 x + \sin^2 x = 1$, por lo que la primera línea (cuyo output no se mostrará por pantalla) asigna a la variable "a" el resultado 1. La segunda línea asigna a la variable "b" la unidad imaginaria, por lo que por pantalla aparecerá el símbolo %i.

3. (2 puntos) El *número áureo* (también llamado *proporción áurea*) se define como la mayor de las raices de $x + 1 = x^2$, dada por

$$\phi = \frac{1 + \sqrt{5}}{2} \simeq 1{,}62$$

Este número representa una relación de tamaños estéticamente afortunada. Dado un rectángulo cuyos lados tienen longitudes a y b, escriba una función en Maxima que reciba dichas longitudes como argumentos, calcule la razón de aspecto del rectángulo (el mayor de los valores a y b, dividido por el menor) y devuelva el cociente entre esa razón de aspecto y el valor del número aúreo almacenado en una variable local, `phi`, de la función.

Solución:

```
cociente(a,b):=block([phi,r],
        phi:(1+sqrt(5))/2,
        r:max(a,b)/min(a,b),
        return(r/phi)
);
```

Programación en C

1. (3 puntos) Escriba la implementación de la función

   ```
   double lever(double c0, double c1, double c2, double *x2);
   ```

   que devuelva los valores x_1 (como valor retornado) y x_2 (en el último argumento) tales que se cumpla el sistema de ecuaciones:

   $$x_1 + x_2 = 1$$
   $$c_1 x_1 + c_2 x_2 = c_0$$

   Solución:

   ```
   double lever(double c0, double c1, double c2, double *x2) {
       double x1;
           x1=(c0-c2)/(c1-c2);
           *x2=1-x1;
       return x1;
   }
   ```

2. (3 puntos) La aproximación numérica de 5 puntos para la derivada primera de una función $f(x)$ tiene la forma

   $$f'(x) \approx \frac{1}{12h}\left[f(x-2h) - 8f(x-h) + 8f(x+h) - f(x+2h)\right]$$

   donde h es una intervalo real pequeño ($h \ll 1$). El error de esta aproximación es, aproximadamente, h^4.

   Escriba una función en C que calcule la derivada de una función $f(x)$ en un cierto punto x_0 con una cierta tolerancia que llamaremos `err`. Consideraremos que la función matemática que queremos derivar, $f(x)$, se ha definido previamente en el programa como por ejemplo:

   ```
   double funcion(double x)
   {
       double y;
           //...
       return y;
   }
   ```

   La función que calcula la derivada debe recibir como argumentos la función matemática que queremos derivar, el punto x_0 en el que queremos obtener la derivada y la tolerancia de la aproximación, `err`. El intervalo h que deberemos emplear estará dado por $h = \texttt{err}^{1/4}$.

   Solución: Este ejercicio admite muchas soluciones, todas ellas válidas. A continuación mostramos tres posibilidades. Hemos incluido la función `main` (que no se pedía en el enunciado del ejercicio) para mostrar cómo se realizaría la llamada a la función pedida, que hemos denominado `derivada()`.

   Solución 1

   ```
   double derivada (double f(double), double x0, double err)
   {
           double h;
           h=pow(err,1./4);
           return 1/(12*h)*(f(x0-2*h)-8*f(x0-h)+8*f(x0+h)-f(x0+2*h));
   ```

```
}

int main(int argc, char** argv)
{
        double x0, err;
        x0=atof(argv[1]);
        err=atof(argv[2]);

        printf("La derivada de la funcion en el punto %g, con una"
               "tolerancia de %g, es %g\n",
               x0, err, derivada(funcion, x0, err));

        return 0;
}
```

Solución 2:

```
double derivada (double (*f)(double), double x0, double err)
{
       double h;
       h=pow(err,1./4);
       return 1/(12*h)*(f(x0-2*h)-8*f(x0-h)+8*f(x0+h)-f(x0+2*h));
}

int main(int argc, char** argv)
{
        double x0, err;
        x0=atof(argv[1]);
        err=atof(argv[2]);

        printf("La derivada de la funcion en el punto %g, con una"
               "tolerancia de %g, es %g\n",
               x0, err, derivada(funcion, x0, err));

        return 0;
}
```

Solución 3: (no usando punteros a funciones)

```
double derivada (double x0, double err)
{
       double h;
       h=pow(err,1./4);
       return 1/(12*h)*(funcion(x0-2*h)-8*funcion(x0-h)
              +8*funcion(x0+h)-funcion(x0+2*h));
}

int main(int argc, char** argv)
{
        double x0, err;
        x0=atof(argv[1]);
        err=atof(argv[2]);
```

```
        printf("La derivada de la funcion en el punto %g, con una"
               "tolerancia de %g, es %g\n",
               x0, err, derivada(x0, err));

        return 0;
}
```