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) Explique de manera breve y razonada qué significado tiene en Maxima el carácter ":".

   Solución: El carácter ":" es, en Maxima, el operador de asignación de un valor a una variable. Es decir, una instrucción de la forma a:b; hará que en adelante cualquier referencia a la variable a sea interpretada como el valor de b.

2. (1 punto) Explique de manera breve y razonada la sintaxis que se emplea en Maxima para escribir una ecuación.

   Solución: En Maxima una ecuación es una condición de igualdad, indicada con el operador "=", entre dos expresiones; por ejemplo, a=b. Para resolver de forma simbólica una ecuación, se emplea la función **solve**, que recibe el primer lugar la ecuación (o sistema de ecuaciones, formado por una lista de igualdades) y en segundo lugar la variable (o lista de variables) que se quiere resolver.

3. (2 puntos) Explique de manera breve y razonada la sintaxis que se emplea en Maxima para definir una función (que reciba uno o varios argumentos, incluyendo la posibilidad de que dicha función emplee ciertas "variables locales").

   Solución: En Maxima una función se define dando su expresión en la forma habitual en matemáticas (formada por el nombre de la función y, entre paréntesis y separados por coma, los nombres dados a sus argumentos) y, tras el operador de asignación de función ":=", la expresión matemática que evalúa la función a partir de sus argumentos. Por ejemplo, la expresión f(x,y):=x^2+y^2; define una función f de dos variables x e y.

   Cuando las operaciones necesarias para evaluar una función requieren pasos más complicados, éstos se pueden incluir en un bloque de instrucciones (función "block") donde al principio se puede declarar una lista de variables que se usarán localmente en los cálculos y, tras ellos, las instrucciones de Maxima que realizan los cálculos necesarios para evaluar la función. El argumento de la instrucción return (o, si no hay ambigüedad, el valor de la última expresión) determina el valor de la función evaluada para los argumentos dados. Por ejemplo, la función

```
g(x,y):=block([n,s],
        s:0,
        for n:1 while n <=length(x) do
                if x[n]=y[n] then s:s+1,
        return(s)
);
```

calcula el número de coincidencias en los valores de las componentes de los vectores (listas) x e y usando variables locales n y s. Obsérvese que en este "bloque" es imprescindible usar la instrucción return(s) ya que si no es ambiguo qué valor debe devolver el cálculo realizado.

Programación en C

1. (2 puntos) Defina una estructura Circulo que guarde las coordenadas del centro de un disco y su radio. Después, escriba una función que reciba dos círculos y devuelva 1 si ambos se superponen o 0 si no lo hacen.

   Solución:

```
struct Circulo {
        float x,y;
        float r;
};

int circulo_super(struct Circulo c1, struct Circulo c2) {
        return (c1.x-c2.x)*(c1.x-c2.x)
              +(c1.y-c2.y)*(c1.y-c2.y) < (c1.r+c2.r)*(c1.r+c2.r);
}
```

2. (4 puntos) En el archivo de texto plano "datos.dat" tenemos N pares de puntos reales (x_i, y_i), $i = 1, \ldots N$, que representan los valores de la función $y(x)$ en el conjunto discreto de valores de x en los que se ha dividido el intervalo $[x_1, x_N]$. Escribir un programa que obtenga la derivada de la función en cada punto y exporte las parejas de datos a un archivo "derivada.dat".

   Para obtener la derivada de un conjunto discreto de puntos consideraremos la aproximación basada en diferencias finitas centradas en el punto:

$$y'(x_i) = \frac{1}{2}\left(\frac{y_i - y_{i-1}}{x_i - x_{i-1}} + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right)$$

   que, como puede observarse, calcula el promedio de las pendientes entre el punto y sus dos vecinos más próximos. En los puntos extremos sólo consideraremos la pendiente con su vecino más próximo:

$$y'(x_1) = \frac{y_2 - y_1}{x_2 - x_1} \qquad y'(x_N) = \frac{y_N - y_{N-1}}{x_N - x_{N-1}}$$

   Solución:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define N 1000

int main(int argc, char** argv) {
        int i, n, npts;
        float x[N], y[N], xx, yy;
        FILE *fin, *fout;

        fin=fopen("datos.dat", "r");
```

```
        npts=0;
        do {
                n=fscanf(fin, "%f %f", &xx, &yy);
                if (n==2) {
                        x[npts]=xx;
                        y[npts]=yy;
                        npts++;
                }
        } while(n==2 && npts<=N);

        fclose(fin);

        fout=fopen("derivadas.dat", "w");

        fprintf(fout,"%g\t%g\n", x[0], (y[1]-y[0])/(x[1]-x[0]) );
        for(i=1; i<npts-1; i++) {
                fprintf(fout,"%g\t%g\n",
                        x[i],
                        0.5*((y[i]-y[i-1])/(x[i]-x[i-1])
                        +(y[i+1]-y[i])/(x[i+1]-x[i])) );
        }
        fprintf(fout,"%g\t%g",
                x[npts-1],
                (y[npts-1]-y[npts-2])/(x[npts-1]-x[npts-2]) );

        fclose(fout);

        return 0;
}
```