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:tan(%pi/4)$
b:sqrt(9/a);
```

Indique el output que se obtiene por pantalla.

Solución: Dado que el símbolo %pi se evalúa al número π, Máxima asignará a la variable "a" el valor de la tangente de $\pi/4$, o sea 1; este valor no se mostrará por pantalla. En la segunda línea se asignará el valor de la raíz cuadrada de 9 (o sea, 3) a la variable "b", y ese valor sí se mostrará en pantalla.

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

```
a:log(%e^3)$
b:sqrt(a);
```

Indique el output que se obtiene por pantalla.

Solución: Máxima evalúa el logaritmo natural del cubo del número e (denotado por %e) como 3 y guarda el resultado en la variable "a"; esta evaluación no se refleja por pantalla. Luego evalúa la raíz cuadrada de "a", lo asigna a la variable "b" y muestra el resultado por pantalla, es decir, $\sqrt{3}$.

3. (2 puntos) Se define la circularidad de una figura plana como el cociente entre su perímetro y el correspondiente a un círculo de igual área. Consideremos un rectángulo con lados de longitudes a y b. Escriba una función en Maxima que, basándose en esta información, calcule en variables locales el perímetro (P) y el área (A) de dicha figura, el radio y el perímetro del círculo de igual área ($R = \sqrt{A/\pi}$ y $P' = 2\pi R$) y que, finalmente, devuelva el cociente entre ambos perímetros (P/P').

Solución:

```
circularidad(a,b):=block([P,A,R,Pc],
        P:2*a+2*b,
        A:a*b,
        R:sqrt(A/%pi),
        Pc:2*%pi*R,
        return(P/Pc)
);
```

Programación en C

1. (3 puntos) Escriba la implementación de las funciones

```
float resSeries(int N, float R[]);
float resParall(int N, float R[]);
```

que devuelven los valores de resistencia total de configuraciones de N resistencias (con valores individuales dados por los R_n) en serie o en paralelo, respectivamente.

Solución:

```
float resSeries(int N, float R[]) {
        int n;
        float Rt=0;
        for(n=0; n<N; n++) {
                Rr+=R[n];
        }
        return R;
}

float resParall(int N, float R[]) {
        int n;
        float iR=0;
        for(n=0; n<N; n++) {
                iR+=1/R[n];
        }
        return 1/iR;
}
```

2. (3 puntos) Supongamos una red cuadrada discreta en la que cada nodo de la red (i, j) con $i = 0, 1, 2, \ldots$ y $j = 0, 1, 2, \ldots$ ha sido definido mediante la estructura:

```
struct punto
{
      int i,j;
};
typedef struct punto Punto;
```

Consideremos ahora que nuestro programa principal calcula dos trayectorias sobre la red descritas mediante sendos vectores de `Puntos` que han sido declarados dentro de `main()` como:

```
Punto camino1[long1];
Punto camino2[long2];
```

Escriba una función que reciba de `main()` estas dos trayectorias con sus respectivas longitudes, y que devuelva el número de puntos de la red que tienen en común.

Solución 1:

```
int compara_caminos(int long_a, Punto *a, int long_b, Punto *b) {
    int n, m, cont;
    cont=0;

    for (n=0; n<long_a; n++) {
```

```
            for (m=0; m<long_b; m++) {
                if (a[n].i==b[m].i && a[n].j==b[m].j) {
                    cont++;
                    /* aquí se podría hacer un break
                       para contar cada punto de corte solo una vez
                       (ambas soluciones son correctas) */
                }
            }
        }

        return cont;
    }
```