En cada ejercicio se evaluará la eficiencia del código, el uso de identificadores significativos, la indentación, escritura correcta de llaves {} y el uso adecuado de la palabra reservada const. Se dispone de dos horas para entregar la prueba y debe realizarse en forma estrictamente individual.

Problema 1: ¿puede formar palíndromo?

Para el desarrollo de un videojuego de palabras se necesita una función que dadas unas letras (un texto), indique si con ellas se puede o no formar al menos un texto palíndromo, pues este hecho puede favorecer o no la puntuación del jugador. Un palíndromo es un texto que se puede leer igual de derecha a izquierda que en el sentido opuesto.

El equipo del videojuego requiere que además de la función, usted provea un pequeño programa que permita probarla. El programa debe leer textos de la entrada estándar, y para cada uno de ellos indicar con un 1 si se puede formar al menos un palíndromo y con 0 lo contrario, como se ve a la derecha. Aunque en el juego no se usan textos más largos de 100 letras, el equipo quisiera poder probar textos de hasta 1000 letras en inglés, todas en minúscula.

1. [5%] Implementa programa de prueba correctamente.
2. [15%] Implementa correctamente función que determina si un texto puede formar palíndromo.
3. [5%] Aplica buenas prácticas de programación.

Ejemplo de entrada:

```
parar
ranarene
soldadosolo
```

Ejemplo de salida:

```
parar: 1
ranarene: 0
soldadosolo: 1
```

Problema 2: ts

1. [5%] ¿Qué imprime en la salida estándar el programa de la derecha?
2. [10%] Rastree la memoria del programa. Su dibujo debe ilustrar el estado del programa cuando el control está por iniciar la ejecución de la línea 20.
3. [5%] Describa qué trabajo realiza la función `ts()`.
4. [5%] ¿Cuáles serían identificadores significativos para las siguientes variables:?
   - `b`
   - `p`
   - `q`
   - `c`
   - `ts`
5. [5% opcional] ¿Qué imprime el programa si la línea 25 se reeplaza por `printf("[%s] [%s]\n", ts(1234567890ull), ts(0));`. Explique rápidamente.

```
1.  #include <stdio.h>
2.
3.  const char* ts(unsigned long long n)
4.  {
5.      static char b[27];
6.      sprintf(b, "%llu", n);
7.
8.      char *p = b;
9.      while ( *p )
10.         ++p;
11.
12.     char* q = p + (p - b - 1) / 3;
13.     for ( short c = 0; p < q; ++c )
14.     {
15.         *q-- = *p--;
16.         if ( c && c % 3 == 0 )
17.             *q-- = ',';
18.     }
19.
20.     return b;
21. }
22.
23. int main()
24. {
25.     printf("[%s]\n", ts(1234567890ull));
26.     return 0;
27. }
```

Problema 3: Gauss-Jordan

La eliminación de Gauss Jordan es un método algebraico para resolver un sistema de *n* ecuaciones con *n* incógnitas. Varios software matemáticos populares realizan este proceso y reportan el resultado, pero no los pasos que realizaron para llegar a él. Estos pasos son muy útiles para estudiantes de cursos de álgebra.

Implemente un programa que realiza la eliminación de Gauss-Jordan e imprime los pasos que aplicó en el proceso. El programa lee de la entrada estándar una matriz aumentada de `n` ecuaciones con `n` incógnitas. El número en la primera fila indica este tamaño *n*. Cada fila representa una ecuación, con los coeficientes las *n* variables y el valor de equivalencia. Por ejemplo la segunda fila en el ejemplo de la derecha representa la ecuación $3x_1 + 5x_2 + 8x_3 = 63$. El programa debe transformar la matriz aumentada en una matriz identidad siguiendo el algoritmo de Gauss-Jordan:

1. Por cada fila (o ecuación) `f` de la matriz:
   a. Hacer el elemento de la diagonal de esa fila $M_{f,f}$ en 1, multiplicando la ecuación completa por el inverso de $M_{f,f}$. Por ejemplo, para la fila `f=1` (línea 1 en el ejemplo de salida) se tiene que $M_{1,1}=3.0$, por lo que todos los coeficientes de la fila `f=1` se dividirán por `3.0` para convertir a $M_{1,1}$ en 1. Esta es la primera operación (`f1 /= 3.0`, impresa en la línea 5 del ejemplo de salida) y el resultado se ve en la segunda matriz impresa entre las líneas 6 a 8 del ejemplo de la derecha.
   b. Por cada una de las restantes filas (o ecuaciones) `g`:
      a. Tomar el opuesto aditivo de $M_{g,f}$, llámese $-M_{g,f}$. Por ejemplo, para la segunda ecuación `g=2` (en la línea 7) se toma $-M_{g,f} = -M_{2,1} = -2.0$ como opuesto aditivo.
      b. Por cada columna `c` de la fila o ecuación `g`:
         a. Sumar al coeficiente $M_{g,c}$ el resultado de multiplicar el opuesto $-M_{g,f}$ con $M_{f,c}$ para convertir a $M_{g,f}$ en `0.0`. Por ejemplo el `0.0` de la segunda ecuación `g=2` en la primera columna `c=1` (línea 13) se obtiene como $M_{g,c} \mathrel{+}= -M_{g,f} * M_{f,c}$, es decir, $M_{2,1} \mathrel{+}= -2.0 * 1.0 \Rightarrow 0.0$. Para la segunda columna `c=3` se tendrá $M_{g,c} \mathrel{+}= -M_{g,f} * M_{f,c} \Rightarrow M_{2,3} \mathrel{+}= -2.0 * 2.66 \Rightarrow 6.0 \mathrel{+}= -2.0 * 2.66 \Rightarrow 6.0 \mathrel{+}= -5.33 \Rightarrow 0.66$ (aparece redondeado como `0.7` en línea 13).

Ejemplo de entrada:

```
1. 3
2. 3 5 8 63
3. 2 3 6 41
4. 1 2 3 24
```

Ejemplo de salida:

```
 1.    3.0   5.0   8.0 |  63.0
 2.    2.0   3.0   6.0 |  41.0
 3.    1.0   2.0   3.0 |  24.0
 4.
 5. f1 /= 3.0:
 6.    1.0   1.7   2.7 |  21.0
 7.    2.0   3.0   6.0 |  41.0
 8.    1.0   2.0   3.0 |  24.0
 9.
10. f2 += -2.0 f1:
11. f3 += -1.0 f1:
12.    1.0   1.7   2.7 |  21.0
13.    0.0  -0.3   0.7 |  -1.0
14.    0.0   0.3   0.3 |   3.0
15.
16. f2 /= -0.3:
17.    1.0   1.7   2.7 |  21.0
18.   -0.0   1.0  -2.0 |   3.0
19.    0.0   0.3   0.3 |   3.0
20.
21. f1 += -1.7 f2:
22. f3 += -0.3 f2:
23.    1.0   0.0   6.0 |  16.0
24.   -0.0   1.0  -2.0 |   3.0
25.    0.0   0.0   1.0 |   2.0
26.
27. f3 /= 1.0:
28.    1.0   0.0   6.0 |  16.0
29.   -0.0   1.0  -2.0 |   3.0
30.    0.0   0.0   1.0 |   2.0
31.
32. f1 += -6.0 f3:
33. f2 += 2.0 f3:
34.    1.0   0.0   0.0 |   4.0
35.    0.0   1.0   0.0 |   7.0
36.    0.0   0.0   1.0 |   2.0
```

En cada paso, se debe imprimir las operaciones sobre las filas que se le están aplicando a la matriz (por ejemplo `f1 /= 3.0` indica que se está dividiendo por `3.0` todos los coeficientes de la fila 1). Tras aplicar el algoritmo a todas las filas, se debe obtener la matriz identidad que encuentra los valores solución al sistema de ecuaciones.

1. [5%] Lee matrices reales correctamente de la entrada estándar.
2. [10%] El programa se puede usar con matrices muy grandes (por ejemplo, mayores a 8MB) sin producir fugas de memoria.
3. [20%] Implementa correctamente la reducción de Gauss-Jordan.
4. [10%] Imprime cada una de las operaciones en las filas.
5. [5%] Imprime el estado de la matriz en cada una de las operaciones.