Pregunta 1

Parte a.- (1 punto) La siguiente función intenta multiplicar las matrices *a* y *b* dejando el resultado en la matriz *c*. Todas las matrices son de 8 x 8. La multiplicación se hace en paralelo en 8 cores.

```
void mult(double **a, double **b, double **c) {
  int i, j, k;
  for (i=0; i<8; i++) {
    if (fork()==0) {
      double *fila= c[i];
      for (j=0; j<8; j++) {
        double s= 0.0;
        for (k=0; k<8; k++)
          s += a[i][k] * b[k][j];
        fila[j]= s;
} } } }
```

Esta función contiene 3 errores de programación relacionados con los procesos de Unix. Indique cuáles son.

Parte b.- (3 puntos) Corrija los errores de la función *mult*. Su función debe usar procesos pesados (*fork*) para paralelizar la multiplicación.

Parte c.- (2 puntos) La siguiente función entrega el número de nodos de una lista enlazada:

```
int largo(Nodo *nodo) {
  int n= 0;
  while (nodo!=NULL) {
    n++;
    nodo= nodo->prox;
  }
  return n;
}
```

Modifique la función *largo* de tal forma que se entregue -1 cuando la lista enlazada contiene un puntero inválido, es decir un puntero a una dirección de memoria no asignada al proceso. Recuerde que si un proceso accede a una dirección inválida, recibirá la señal *SIGSEGV* (*segmentation violation* o *segmentation fault*). Ud. necesitará usar variables globales y definir funciones adicionales.

Pregunta 2

Programe el servidor y el cliente de un sistema para rematar antigüedades por Internet. Un solo vendedor remata una de estas reliquias invocando el comando *remato*, que corresponde al servidor. Los múltiples interesados invocan el comando *compro* para ofrecer un precio por la antigüedad, especificando el nombre y el precio que están dispuestos a pagar. Se adjudicará la reliquia al comprador que ofrezca el mayor precio. El siguiente ejemplo de uso muestra la funcionalidad requerida:

Vendedor	*Comprador 1*	*Comprador 2*	*Comprador 3*
% **remato** *(espera)*			
Pedro ofrece 3	% **compro Pedro 3** *(espera)*		
Juan ofrece 5	*(termina)* Juan ofrece 5 %	% **compro Juan 5** *(espera)*	
			% **compro Diego 4** Juan ofrece 5 %
Diego ofrece 8		*(termina)* Diego ofrece 8 %	% **compro Diego 8** *(espera)*
terminar vendida a Diego en 8 %			*(termina)* adjudicada %

Note que el *prompt %* indica cuando un comando termina o si debe esperar. El tiempo avanza hacia abajo. En **negritas** aparece lo que escribió un usuario directamente. Entre paréntesis y letra *cursiva* se enfatiza cuando un comando debe esperar o terminar. El resto es la salida de alguno de los comandos.

En un instante dado solo puede haber un comprador en espera y debe ser aquel que ofrece el mayor precio. Para el resto de los compradores su oferta fue rechazada. El vendedor adjudica la antigüedad escribiendo una línea que diga ``*terminar*'' en la entrada estándar del comando *remato*.

Requerimientos: Los comandos *remato* y *compro* corren en la misma máquina y usan el puerto 3000 para comunicarse. En el servidor use *select* para determinar si debe aceptar un nuevo cliente (*j_accept*) o leer la entrada estándar. Si la línea ingresada por el vendedor no es exactamente ``*terminar*'', Ud. debe ignorarla.

Observe que no necesita crear un thread o proceso por cada cliente que se conecta. Puede ser optimista: el puerto siempre estará libre, la conexión siempre es exitosa, *select*, *read* y *write* no fallan, etc.