Pregunta 1

Se ha programado secuencialmente la función *suma* de la siguiente manera:

```
int suma(double x0, double dx, int n, double *pres) {
  double s= 0;
  for (int k= 0; k<n; k++)
    s+= f(g(x0+dx*k));
  *pres= s;
  return 0;
}
```

Cada evaluación de *f* y *g* es lenta y por eso se requiere paralelizar.

Parte a.- Reescriba esta función paralelizándola para una máquina dual-core. Para ello use una sola vez la llamada al sistema *fork*. El proceso hijo debe realizar todas las evaluaciones de la función *g*, enviando sus resultados al padre por medio de un *pipe*. El padre realiza todas las evaluaciones de la función *f* tomando como argumento los resultados calculados por el hijo. El resultado final retornado por la función *suma* debe ser el mismo calculado por su versión original. No olvide enterrar adecuadamente al hijo.

Parte b.- Se sabe que de vez en cuando se realiza una división por cero al evaluar la función *f* lo cual gatilla la señal SIGFPE terminando el proceso completo. Modifique su versión de la parte a de manera que cuando ocurre la división por cero (solo puede ocurrir en el proceso padre) la función *suma* retorne -1 en vez de terminar el proceso. De ocurrir la división por cero, además Ud. debe matar al proceso hijo para que al enterrarlo no tenga que esperar a que el hijo haga todas las evaluaciones de *g*.

Observaciones: Ud. deberá escribir una función de atención para la señal SIGFPE, la que no puede simplemente retornar porque si lo hace entonces volverá a ocurrir la división por cero, gatillando nuevamente la señal SIGFPE. Escriba una versión de suma para la parte a y otra para la parte b. No intente escribir una sola versión para ambas partes porque los errores que cometa en la parte b le restarán puntaje en ambas partes.

Pregunta 2

Se requiere programar el servidor y el cliente de un sistema que permite formar un grupo de 4 personas. El servidor se llama *grupo*, corre siempre en *localhost* y escucha a los clientes en el puerto 3000. No recibe parámetros.

El cliente se llama *miembro* y recibe como único parámetro el nombre de la persona que será miembro del grupo. Este comando solo debe terminar cuando se formó el grupo completo, entregando los nombres de todos los integrantes del grupo. A partir de la llegada del quinto miembro se muestra un mensaje de error. El siguiente es un ejemplo de uso del comando *miembro*:

Pedro	***Juan***	***Diego***	***Alberto***
$ **miembro pedro** *(espera)*			
	$ **miembro juan** *(espera)*		
		$ **miembro diego** *(espera)*	
pedro juan diego alberto $	pedro juan diego alberto $	pedro juan diego alberto $	$ **miembro alberto** pedro juan diego alberto $
$ **miembro tomas** Lo siento, el grupo ya fue completado $			

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.

Requerimientos: Programe el cliente y el servidor de este sistema. El cliente debe reproducir exactamente la salida que se muestra en el ejemplo, exceptuando el texto que dice *(espera)*. En el servidor: use threads para atender los clientes; no programe la función *main*; programe la función de servicio (*serv*); declare las variables globales que necesita señalando como deben ser inicializadas; no necesita preocuparse por el término del servidor.