Pregunta 1

Parte a.- 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.

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.- Programe la función:

```
char *ultimaDireccionValida(char *ptr);
```

Esta función debe entregar la última dirección válida que se puede leer a partir de *ptr*. Para calcularla lea el caracter apuntado por *ptr* e incremente *ptr* en 1. Repita esta lectura e incremento indefinidamente hasta que se produzca el *segmentation fault*. Capture la señal *SIGSEGV*. Resguarde *ptr* en una variable global. El último valor de *ptr* menos 1 es la última dirección válida.

Pregunta 2

Se requiere programar el servidor y el cliente de un sistema para hacer colectas por Internet. El servidor se llama *colecta*, corre siempre en *localhost* y escucha a los clientes en el puerto 3000. Recibe como argumento el monto de la meta que se debe alcanzar. Ejemplo de invocación:

```
$ ./colecta 11
```

El cliente se llama *aportar* y recibe como argumento el monto del aporte. El cliente debe esperar hasta que se haya alcanzado la meta. En tal caso entrega el monto efectivo aportado. Por ejemplo si faltan 3 euros para alcanzar la meta y una persona aporta 5, entonces aportar despliega 3. Si una persona aporta cuando la meta ya fue alcanzada, entonces se despliega 0 de inmediato. El siguiente es un ejemplo de uso. La meta a recaudar es 11.

Persona 1	***Persona 2***	***Persona 3***	***Persona 4***
$ **./aportar 5** *(espera)*			
	$ **./aportar 2** *(espera)*		
5 $ *(termina)*	2 $ *(termina)*	$ **./aportar 7** 4 $	
			$ **./aportar 3** 0 $

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 de colectas. El cliente debe reproducir exactamente la salida que se muestra en el ejemplo, exceptuando el texto que dice *(espera)* o *(termina)*. En el servidor: use threads para atender los clientes; no programe la función *main*; programe la función de servicio (*serv*); suponga que la meta a recaudar se encuentra en una variable global; deberá usar otras variables globales; señale cómo se inicializan en el *main*.