Las acciones de la empresa ACME se transan en una bolsa de comercio cuyos operadores son representados mediante threads. Para comprar o vender una acción los operadores usan las funciones *compro* y *vendo*. Sus encabezados son:

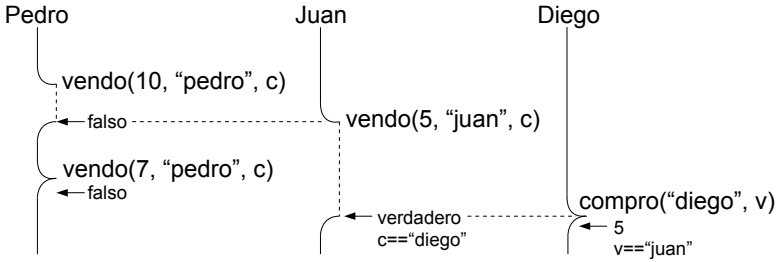
```
int vendo(int precio, char *vendedor, char *comprador);
int compro(char *comprador, char *vendedor);
```

En el cuadro siguiente se muestra a la izquierda el código usado por los múltiples threads vendedores de acciones y a la derecha el código de los múltiples compradores.

```
char *nom= miNombre();
int precio= miPrecio();
char comprador[100];
if (vendo(precio, nom,
          comprador)) {
  printf("vendi a %s\n",
         comprador);
}
```

```
char *nom= miNombre();
char vendedor[100];
sleep(tiempoAleatorio());
int precio= compro(nom, vendedor);
if (precio>0) {
  printf("compre a %s en %d\n",
         vendedor, precio);
}
```

La función *compro* transa una sola acción con el vendedor más barato de ese momento, retornando el precio pagado y copia el nombre del vendedor en el *2do parámetro*. Si en ese momento no hay ningún vendedor el precio será 0 y *compro* retorna de inmediato. La función *vendo* ofrece una acción al precio indicado y espera hasta que (i) aparezca un comprador, en cuyo caso retorna verdadero y copia el nombre del comprador en el 3er parámetro, o (ii) aparezca (o ya hay) un vendedor con un precio menor, retornando falso en tal caso. El siguiente diagrama muestra un ejemplo de ejecución.



Pedro llama a *vendo*, que retorna falso cuando Juan llama a *vendo* con un precio menor. La segunda llamada de Pedro fracasa de inmediato porque su precio todavía es mayor al de Juan. Juan sí tiene éxito cuando Diego llama a *compro* y por lo tanto la llamada a *vendo* de Juan retorna verdadero, copiando el nombre "diego" en el parámetro *comprador*. Por su parte *compro* retorna 5 (el precio) y copia "juan" en el parámetro *vendedor*.

Programe las funciones *vendo* y *compro*. Para la sincronización debe usar un mutex y **una sola condición**. Observe que nunca habrá más de un vendedor en espera. Use variables globales. Está prohibido usar *busy-waiting*.

Instrucciones

Baje *t2.zip* de U-cursos y descomprímalo. El directorio *T2* contiene los archivos *test-bolsa.c*, *Makefile*, *bolsa.h* (con los encabezados requeridos) y otros archivos. Ud. debe programar en el archivo *bolsa.c* las funciones *compro* y *vendo*.

Pruebe su tarea bajo Debian 12 de 64 bits. Ejecute el comando *make* sin parámetros. Le mostrará las opciones que tiene para compilar su tarea. Estos son los requerimientos para aprobar su tarea:

- *make run* debe felicitarlo por aprobar este modo de ejecución. El *speed up* reportado debe ser de al menos 1.2.
- *make run-g* debe felicitarlo.
- *make run-san* debe felicitarlo y sanitize no debe reportar ningún tipo de problema.

Invoque el comando *make zip* para ejecutar todos los tests y generar un archivo *bolsa.zip* que contiene *bolsa.c*, con su solución, y *resultados.txt*, con la salida de *make run*, *make run-g* y make *run-san*.

Entrega

Ud. solo debe entregar por medio de U-cursos el archivo *bolsa.zip* generado por *make zip*. Recuerde descargar el archivo que subió, descargar nuevamente los archivos adjuntos y volver a probar la tarea tal cual como la subió a U-cursos. Solo así estará seguro de no haber entregado archivos incorrectos. Se descuenta medio punto por día de atraso. No se consideran los días de receso, sábado, domingo o festivos.