EXAMEN SCD P3 2020 - 2021

1. Construir un programa MPI, que se deberá llamar prod_cons_ex.cpp, con operaciones de paso de mensaje síncronas siguiendo el esquema del productor-consumidor, en el que tenemos cuatro procesos consumidores (identificadores del 0 al 3) y seis procesos productores (identificadores del 4 al 9) que producirán 15 elementos cada uno. Ambos tipos de procesos se comunican mediante un proceso intermedio "buffer" con identificador 10. Dicho proceso intermedio gestiona un vector de tamaño 10. Otras consideraciones:

   - La gestión del vector se debe hacer en modo LIFO, es decir, se debe consumir el último elemento producido.

   - El proceso buffer trata de equilibrar la atención a productores y consumidores independientemente de la ocupación del vector. Por tanto, cada vez que atiende a cuatro productores de forma consecutiva (sin atender entre medias a ningún consumidor), no atiende más peticiones de los productores hasta que no haya atendido a un proceso consumidor.

2. Un food-truck está aparcado cerca de la ETSIIT y sólo vende bocatas de panceta con morcilla untada. Es frecuentado por 12 alumnos de SCD poco cuidadosos con su alimentación (procesos 0 al 11). Los alumnos con identificador par compran un bocata en cada iteración y los alumnos con identificador impar son más "tragones" y compran dos bocatas en cada iteración. En cada iteración, los alumnos mandan un mensaje al proceso dependiente (rank 12) con etiquetas distintas según el número de bocatas a comprar. Si hay menos de 3 bocatas en el camión, el proceso dependiente solo acepta peticiones de un bocata. Cuando no quedan bocatas, el proceso dependiente manda un mensaje al proceso cocinero (rank 13), que se encarga de preparar en el camión 20 nuevos bocatas. Suponer que el camión tiene inicialmente 20 bocatas, considerar todos los procesos como bucles infinitos e implementar dicho programa en MPI con operaciones de paso de mensaje síncronas y mensajes para seguir la traza del programa (el archivo se deberá llamar "ejerciciop2.cpp").

```
Proceso_alumno
   {retardo aleatorio ("camino hacia el camion")}
   s_send(dependiente,var) {peticion_bocata/s}
   receive(dependiente, var)  {bocata/s servido}
   {retardo aleatorio ("comer mucha grasa")}
```

```
Proceso_cocinero
    receive(dependiente,var) {preparado para preparar bocatas}
    {retardo aleatorio (preparar bocatas en en camión)}
    ssend(dependiente, var) {bocatas preparados}
```

```
Proceso_dependiente
   Si quedan en la tienda 3 o más bocatas
       recibir un mensajes de cualquier proceso alumno
   En caso contrario, si queda al menos una bocata
       recibir un mensaje de alumnos que sólo piden un bocata
   En caso contrario, (no hay bocatas en el camión)
        s_send(cocinero,var)
        recibir un mensaje del cocinero
   emisor:= proceso emisor del mensaje recibido anteriormente
   Si  emisor es un alumno
       actualizar estado {menos bocatas en la tienda}
       s_send(emisor,var) {bocata servido}
   En caso contrario (se ha recibido un mensaje del cocinero)
       actualizar estado {bocatas disponibles}
```