UNIVERSIDAD DE MÁLAGA | uma.es

Dpto. de Lenguajes y Ciencias de la Computación

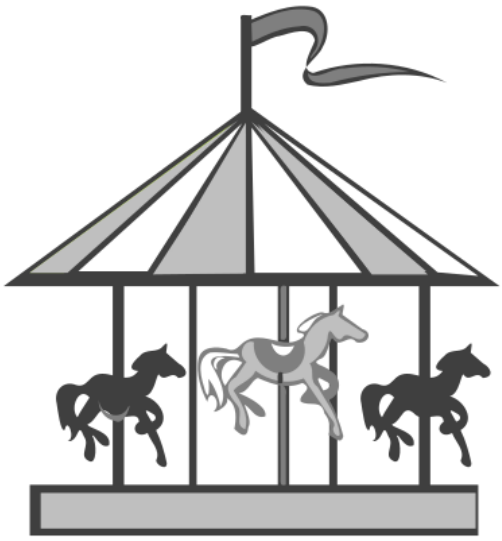
Programación de Sistemas y Concurrencia

Examen Junio 2019

APELLIDOS_____NOMBRE_____

DNI _____ ORDENADOR _____GRUPO/TITULACIÓN_____

Bloque de Concurrencia (6 ptos)



El tiovivo de la feria de Málaga tiene capacidad para 5 pasajeros, cada uno ocupa un caballito de la atracción.

El operario de la atracción se encarga de parar el tiovivo para que se bajen los pasajeros. Cuando no quedan más pasajeros, da paso a nuevos pasajeros que estaban esperando en la cola. Cuando todas las plazas están ocupadas, el operario pone en marcha el tiovivo para que todos disfruten del paseo.

Los pasajeros esperan en la cola hasta que la atracción está parada y vacía para poder subir. Una vez que conseguido un caballito, permanecen en la atracción hasta que termina el paseo y el operario ha parado la atracción.

Se pide implementar dos soluciones de este sistema, una basada en semáforos binarios y otra en locks o métodos sincronizados. Para ello se proporcionan las siguientes clases:

- Principal: Crea e inicializa los diferentes componentes del sistema.
- Pasajero: modela el comportamiento de un pasajero utilizando los métodos proporcionados por el recurso compartido Tiovivo.
- Operario: modela el comportamiento del operario del tiovivo utilizando los métodos proporcionados por el recurso compartido Tiovivo.
- Tiovivo (Semáforos binarios 3 ptos / Locks o sincronizados 3 ptos): modela el recurso compartido y proporciona los siguientes métodos:
    - `public void subir(int id):` método usado por un pasajero que quiere subir a la atracción. Un pasajero no puede subirse hasta que el tiovivo está parado, no haya pasajeros del paseo anterior y haya espacio.
    - `public void bajar(int id):` método usado por un pasajero que quiere bajar de la atracción. Un pasajero no puede bajar de la atracción hasta que no dé un paseo y el tiovivo haya parado.
    - `public void esperaLleno():` método usado por el operario para que los pasajeros que estaban en la cola puedan subirse hasta completar todas las plazas. Cuando no quedan plazas libres el tiovivo se pone en marcha.
    - `public void finViaje():` método usado por el operario parar el tiovivo y dar paso a que los pasajeros que estaban dentro puedan bajarse.

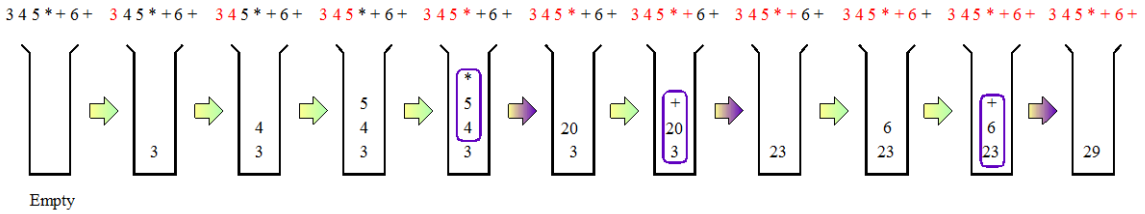
Notas:

1. La única documentación que se puede utilizar es el API de Java disponible en Eclipse.
2. Hay que utilizar los ficheros fuentes/proyecto que se proporcionan
3. Hay que subir al campus virtual (del grupo) la implementación de la clase Tiovivo.

Bloque de C (3 ptos)

Una calculadora en Notación Polaca Inversa (*Reverse Polish Notation*, RPN) permite hacer cálculos aritméticos de expresiones utilizando notación posfija. De esta manera, una expresión como 3+4*5+6 se representaría en RPN de la siguiente manera:
3 4 5 * + 6 +

Para calcular los resultados se utiliza una pila en la que se introducen los operandos (números). Tan pronto como se lee un operador (sólo los operadores binarios +, -, * y /), éste toma como operandos a los dos números que se encuentren en la cima de la pila, los elimina y los sustituye por el resultado de operar con ellos. Como ejemplo, los cálculos anteriores producen que la pila evolucione de la siguiente manera:



donde el resultado final es el último operando que queda en la pila al final, o sea, 29. Es importante darse cuenta de que aunque en el dibujo los operadores aparecen en la pila, estos en realidad no se insertan sino que se utilizan para evaluar los dos últimos elementos de la pila, e insertar el resultado.

Deben implementarse las siguientes funciones en el fichero fuente Stack.c:

```
// Crea una pila vacía.
T_Stack create();

// Devuelve true si la pila está vacía y false, en caso contrario.
int isEmpty(T_Stack q);

// Inserta un número en la pila.
void push(T_Stack * pq, int operando);

// Realiza la operación indicada por el operador sobre los dos
// ultimos elementos de la cima e inserta
// el resultado de la operación.
// Devuevle true si todo va bien o false en caso contrario.
int pushOperator(T_Stack * pq, char operator);

// Devuelve en dato el número que hay en la cima de pila
// y lo elimina de ésta.
// Devuelve true si todo va bien o false en caso contrario.
int pop(T_Stack * pq, int * dato);

// Libera la memoria de la pila y la deja vacía.
void destroy(T_Stack * pq);
```

Además, en el fichero Driver.c debe implementarse la siguiente función:

```
int process(char * filename)
```

Esta función toma como parámetro el nombre de un fichero de texto que contiene una expresión en RPN (cada línea contiene sólo un operador o un operando) y devuelve el resultado de evaluarla (como el

fichero es de texto, se puede ver el formato con cualquier editor). Se proporciona el fichero `source.calc` con un ejemplo de contenido de dicho tipo de ficheros de entrada.

Para facilitar la implementación de esta función, se proporciona el método de ayuda:

```
int text2Int(char * text);
```

que convierte un texto formado por dígitos en su equivalente numérico.

Anexo. Los prototipos de las funciones de lectura y escritura binaria en ficheros de la biblioteca `<stdio.h>` son los siguientes (se dan por conocidos los prototipos de las funciones de `<stdlib.h>` que necesites, como `free` o `malloc`):

```
FILE *fopen(const char *path, const char *mode);
```

Abre el fichero especificado en el modo indicado ("rb" para lectura binaria y "wb" para escritura binaria). Devuelve un puntero al manejador del fichero en caso de éxito y NULL en caso de error.

```
int fclose(FILE *fp);
```

Guarda el contenido del buffer y cierra el fichero especificado. Devuelve 0 en caso de éxito y -1 en caso de error.

```
char *fgets(char *str, int num, FILE *stream);
```

Lee del fichero apuntado por `stream` como máximo `num-1` caracteres o hasta que se alcanza una nueva línea o el final del fichero y devuelve la cadena de caracteres correspondiente.

```
int fgetc (FILE *stream);
```

Devuelve el carácter siguiente (si está presente) como un unsigned char convertido a int, desde el fichero apuntado por `stream`, y avanza el indicador de posición de ficheros.