Paradigmas de la Programación
Segundo Parcial

1. Modifique la función eager `Quicksort` para obtener `LQuicksort` que sea lazy e "incremental". Esto significa que si pedimos los primeros elementos de la lista la función no ordenará todos los elementos. Describa brevemente y en palabras el funcionamiento de una ejecución de la función `LQuicksort`.

   Ayuda: Tendrá que modificar/cambiar algunas funciones que usa el quicksort. El Append es el de la sección 3.4.2.1 (es eager). Resta puntos modificar funciones innecesariamente.

```
declare
proc {Partition L2 X L R}
   case L2
   of Y|M2 then
      if Y<X then Ln in
         L=Y|Ln
         {Partition M2 X Ln R}
      else Rn in
         R=Y|Rn
         {Partition M2 X L Rn}
      end
   [] nil then L=nil R=nil
   end
end

declare
fun {Quicksort L}
   % Usar primer elemento de L como pivot.
   case L of X|L2 then Left Right SL SR in
      {Partition L2 X Left Right}
      SL={Quicksort Left}
      SR={Quicksort Right}
      {Append SL X|SR}
   [] nil then nil end
end

% Ejemplo: mostrar el tercer elemento de la lista ordenada.
declare
Xs={Quicksort [5 8 4 7 3 2 7 6 0 5 4 1 0 4]}
{Browse Xs.2.2.1}
```

2. El siguiente código es una solución al primer servidor pedido en el ejercicio 3 del práctico 8. Lee de un puerto `P` mensajes de la forma `Id#Op` a donde `Id` es un número y `Op` es `add` o `mul`. Por cada mensaje `Id#Op` recibido, suma o multiplica (según `Op`) el `Id` por 11 y escribe en el stream `Xs` el elemento `Id#Op#Resultado`.

```
declare Xs S P
proc {Servidor1}
   P = {NewPortObject Xs
        fun {$ S M} Snew in
           case M of Id#add then S = Id#add#(Id+11)|Snew
           [] Id#mul then S = Id#mul#(Id*11)|Snew end
           Snew
        end}
end
```

(Importante: La función `NewPortObject` es la de la página 351 del libro.)

a) Defina un procedimiento `Cliente` que tome un `Id` y una lista de operaciones (`add` o `mul`), que en un thread aparte envíe al servidor las operaciones especificadas en la lista.

b) Modifique el servidor de manera que vaya sumando los resultados de todas las operaciones efectuadas y tambien los vaya escribiendo en el stream `Xs` con la forma `Id#Op#Resultado#Suma`.

c) Escriba el código que ejecuta el servidor, dos clientes con IDs 1 y 2 que envían `[add mul add mul]` y `[mul add add]` resp., y un browser que muestra la salida del stream.

40%

3. Durante el curso se introdujo el concepto de puerto, que es un canal simple de comunicación. Los puertos tienen las operaciones `{NewPort S P}`, que retorna un puerto `P` con stream `S`, y `{Send P X}`, que manda un mensaje X en el puerto P. De estas operaciones, es claro que los puertos son ADTs con estado. Para este ejercicio implemente puertos como ADT utilizando celdas. El ADT que implemente debe ser abierto, seguro, y empaquetado.

estado explícito

33%

4.
```
fun {From C1 C2 C3}
        c(methods:M1 attrs:A1)={Unwrap C1}
        c(methods:M2 attrs:A2)={Unwrap C2}
        c(methods:M3 attrs:A3)={Unwrap C3}
        MA1={Arity M1}
        MA2={Arity M2}
        MA3={Arity M3}
        ConfMeth={Minus {Inter MA2 MA3} MA1}
        ConfAttr={Minus {Inter A2 A3} A1}
in
        if ConfMeth\=nil then
                raise illegalInheritance(methConf:ConfMeth) end
        end
        if ConfAttr\=nil then
                raise illegalInheritance(attrConf:ConfAttr) end
```