Parallel computing

Solution of exercise on basic communication primitives

- *With one-sided communication (BSP)*

```
init(myproc_id,p)
bsp_push_reg(storeL)  ← registration primitive
bsp_push_reg(storeR)  ← registration primitive
bsp_put(infoR, storeL, (myproc_id+1)mod p)
bsp_put(infoL, storeR, (myproc_id-1)mod p)
bsp_sync()
```

- *With two-sided blocking non-buffered communication (even number of processors assumed)*

```
initcomm (myproc_id,p)
label = null    (parameter label is not really used)
if (myproc_id is even)
   then  send (infoR, (myproc_id+1)mod p, label)
         receive (storeR, sendproc, labelout)
         if (sendproc ≠ (myproc_id+1)mod p then fout !
   else
         receive (storeL, sendproc, labelout)
         if (sendproc ≠ (myproc_id-1)mod p then fout !
         send (infoL, (myproc_id-1)mod p, label)
endif
if (myproc_id is oneven)
   then  send (infoR, (myproc_id+1)mod p, label)
         receive (storeR, sendproc, labelout)
         if (sendproc ≠ (myproc_id+1)mod p then fout !
   else
         receive (storeL, sendproc, labelout)
         if (sendproc ≠ (myproc_id-1)mod p then fout !
         send (infoL, (myproc_id-1)mod p, label)
endif
```

- *With two-sided blocking buffered communication*

```
init(myproc_id,p)
send(infoR, (myproc_id+1)mod p, 'right')
send(infoL, (myproc_id-1)mod p, 'left')
for i = 1 to 2
    receive(temp, fromproc, label)
    if (label=right)
         then  storeL = temp
         else  storeR = temp
    endif
endfor
```

Remark:
The test `if (label=right)` can be replaced by
`if (fromproc = myproc_id+1).`