# Appendix C: Sample programs used for additional testing

```
./2dautomata.occ
SEQ
    [2][2][2]INT rule:
    rule [0][0][0] := 0
    rule [0][0][1] := 1
    rule [0][1][0] := 1
    rule [0][1][1] := 1
    rule [1][0][0] := 1
    rule [1][0][1] := 0
    rule [1][1][0] := 0
    rule [1][1][1] := 0
    [32]INT yCoord:
    [32]INT me:
    SEQ i = [0 FOR 31]
        me[i] := 0
    me[16] := 1
    [32]INT myLeft:
    [32]INT myRight:
    [32]CHAN OF INT left:
    [32]CHAN OF INT right:
    PAR
        SEQ
            yCoord[0] := 0
            WHILE yCoord[0] < 32
                SEQ
                    GRAPHICS[0][yCoord[0]] ! me[0]+7
                    left[0] ! me[0]
                    left[31] ? myLeft[0]
                    right[31] ! me[0]
                    right[0] ? myRight[0]
                    me[0] := rule[myLeft[0]][me[0]][myRight[0]]
                    yCoord[0] := yCoord[0] + 1
        PAR i = [1 FOR 31]
            SEQ
                yCoord[i] := 0
                WHILE yCoord[i] < 32
                    SEQ
                        GRAPHICS[i][yCoord[i]] ! me[i]+7
                        left[i-1] ? myLeft[i]
                        left[i] ! me[i]
                        right[i] ? myRight[i]
                        right[i-1] ! me[i]
                        me[i] := rule[myLeft[i]][me[i]][myRight[i]]
                        yCoord[i] := yCoord[i] + 1
./ring.occ
SEQ
    [32]INT me:
```

```
    SEQ i = [0 FOR 31]
        me[i] := 0
    me[16] := 1
    [32]INT myLeft:
    [32]CHAN OF INT left:
    PAR
        SEQ
            left[0] ! me[0]
            left[31] ? myLeft[0]
        PAR i = [1 FOR 31]
            SEQ
                left[i-1] ? myLeft[i]
                left[i] ! me[i]
./slowprime.occ
---
filter (\x -> x == 0) = 0
---

SEQ
    INT x:
    x := 37
    CHAN chan:
    PAR i = [2 FOR 37]
        SEQ
            INT c:
            c := x
            WHILE c >= 0
                c := c - i
            SERIAL ! c
./countdownup.occ
---
filter xs (\x -> x > 0) == sort (filter xs (\x -> x > 0))
filter xs (\x -> x < 0) == reverse (sort (filter xs (\x -> x < 0)))
...Or something
---
SEQ
    CHAN OF INT left:
    CHAN OF INT right:
    INT x:
    INT y:
    INT z:
    x := 1
    y := 1
    PAR
        WHILE TRUE
            SEQ
                right ! x
                x := (x+1)
        WHILE TRUE
```

```
            SEQ
                left ! y
                y := (y-1)
        WHILE TRUE
            ALT
                TRUE & left ? z
                    SERIAL ! z
                right ? z
                    SERIAL ! z
./no_arrays_ring.occ
SEQ
    INT x:
    INT y:
    INT z:
    x := 0
    y := 1
    z := 2
    INT xl:
    INT yl:
    INT zl:
    CHAN OF INT xleft:
    CHAN OF INT yleft:
    CHAN OF INT zleft:
    PAR
        SEQ
            yleft ! x
            xleft ? xl
        SEQ
            yleft ? yl
            zleft ! y
        SEQ
            zleft ? zl
            xleft ! z

./sameamount.occ
---
length (filter (\x -> x == 1) xs) == 5
length (filter (\x -> x == 0) xs) == 5
---

SEQ
    INT x:
    CHAN chan:
    PAR
        SEQ i = [0 FOR 4]
            chan ! 1
        SEQ i = [0 FOR 4]
            chan ! 1
        WHILE TRUE
```

```
            SEQ
                chan ? x
                SERIAL ! x
./alternate.occ
---
xs = [0,1,0,1..]
---
SEQ
    INT x:
    CHAN chan:
    PAR
        WHILE TRUE
            SEQ
                chan ! 0
                chan ! 1
        WHILE TRUE
            SEQ
                chan ? x
                SERIAL ! x
```