Input/Output

- Basic printing
- Character and item repeaters and consumers
- Simple graphics output

Basic Printing

help io

`=>`	print arrow
`==>`	pretty print arrow
`pr(<item>)`	print item
`ppr(<item>)`	pretty print item
`spr(<item>)`	print item then a space
`npr(<item>)`	print item then a newline
`sp(<integer>)`	print <integer> spaces
`nl(<integer>)`	print <integer> newlines

```
3->n;
pr(n); pr('Bottles'); sp(5); pr(n+1); nl(1); pr('hanging');
3Bottles     4
hanging
```

Character Manipulation — 1

```
define cleanup(file1, file2, oldchar, newchar);
    ;;; read all the CHARACTERS from file1 to file2
    ;;; replacing any instances of oldchar by newchar
    lvars inchar outchar thischar;
    discin(file1) -> inchar;
    discout(file2) -> outchar;
    until thischar = termin do
        inchar() -> thischar;
        if thischar = oldchar
        then newchar -> thischar
        endif;
        outchar(thischar)
    enduntil
enddefine;
```

Character Manipulation — 2

io1.p =

```
5 green bottles hanging on the wall
```

```
cleanup('io1.p', 'io2.p', '\s', '\n');
```

io2.p =

```
5
green
bottles
hanging
on
the
wall

```

```
cleanup('io1.p', 'io3.p', 'e', '*');
```

io3.p =

```
5 gr**n bottl*s hanging on th* wall
```

Item Manipulation — 1

```
define censor(file1, file2, olditem, newitem);
    ;;; read all the ITEMS from file1 to file2
    ;;; replacing any instances of olditem by newitem
    lvars initem outitem thisitem;
    incharitem(discin(file1)) -> initem;
    outcharitem(discout(file2)) -> outitem;
    until thisitem = termin do
        initem() -> thisitem;
        if thisitem = olditem
        then newitem -> thisitem
        endif;
        outitem(thisitem)
    enduntil
enddefine;
```

Item Manipulation — 2

io1.p = $\boxed{\text{5 green bottles hanging on the wall}}$

```
censor('io1.p', 'io4.p', 5, 17);
```

io4.p = $\boxed{\text{17greenbottleshangingonthewall}}$

```
censor('io1.p', 'io4.p', "green", "blue");
```

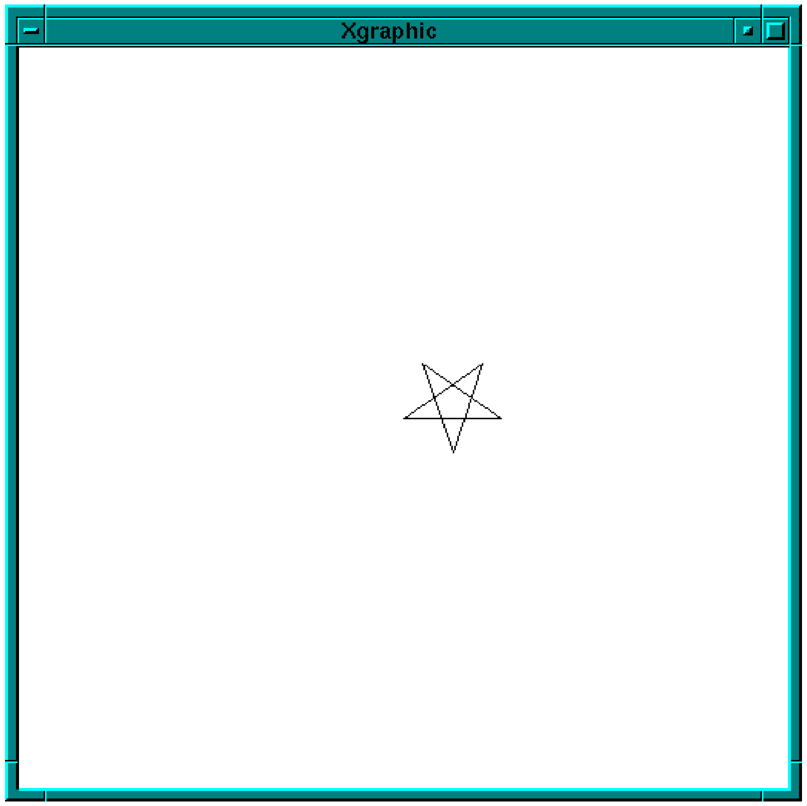
io5.p = $\boxed{\text{5bluebottleshangingonthewall}}$

Using Graphics Packages in POP-11

help rc_graphic

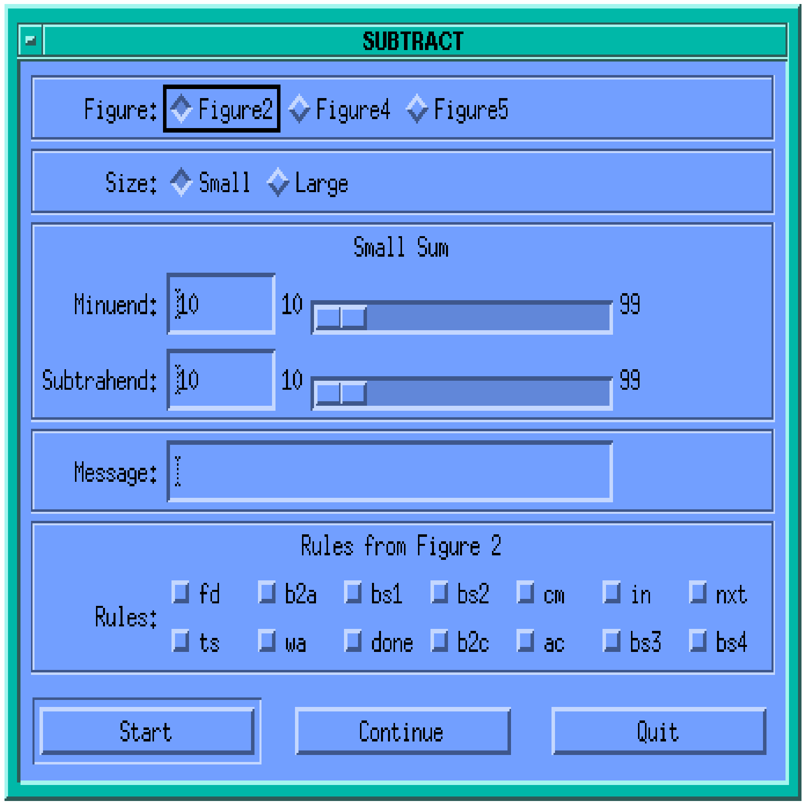
```
lib rc_graphic;

define star(side);
  repeat 10 times rc_draw(side); rc_turn(144) endrepeat
enddefine;

rc_start();
star(63);
```

Graphical Output



teach rc_graphic

Graphical Input



SUBTRACT

Figure: [✓ Figure2] Figure4 Figure5

Size: Small Large

Small Sum

Minuend: 10 10 99

Subtrahend: 10 10 99

Message:

Rules from Figure 2

Rules: fd b2a bs1 bs2 cm in nxt ts wa done b2c ac bs3 bs4

teach propsheet

GUI Building

```
uses bhamlib;
<ENTER> teach rclib
```