Problem 1: (20 points)

Write a program that opens a 300 x 300 window and permits the user to draw circles of varying size and color...

```pascal
procedure main()
  WOpen("size=300,300","drawop=reverse")
  colors := create [WAttrib("fg="||!="red","green","blue")]
  @colors
  repeat {
    case Event() of {
      &rpress: @colors
      &lpress: {
        x := &x
        y := &y
        r := 1
        DrawCircle(x,y,r)
        until (c := Event()) === "." do {
          newr := r
          case c of {
            "+": newr := r + 1
            "-": newr := r - 1
          }
          if newr ~= r then {
            DrawCircle(x,y,r)
            DrawCircle(x,y,r := newr)
          }
        }
      }
    }
  }
end
```

Mr. Pawlowski had a very interesting solution for the radius increment/decrement. Here is the essence of it:

```pascal
case e := Event() of {
  !"+-": radius := e(radius, 1)
}
```

Mr. Wampler used co-expressions as co-routines to call between two procedures, place() and size(), to handle the switching between modes.
Problem 2: (20 points)

Write a procedure `FillGizmo(x, y, cap, stem, inset)` that draws a gizmo at the coordinates `(x, y)`.

```plaintext
procedure FillGizmo(x, y, cap, stem, inset)
  FillCircle(x+cap, y+cap, cap, 0, -&pi)
  FillCircle(x+cap, y+cap+stem, cap, 0, &pi)
  FillRectangle(x+inset, y+cap, (cap-inset)*2, stem)
return
end
```

Coordinate translation with `dx/dy` could be used but in this case it seemed easier to manually offset than to save/set/restore `dx` and `dy`.

Problem 3: (12 points)

Write a procedure `format(fmt, v[])` that does simple `printf`-like formatting of the values in `v` based on the specifications in the string `fmt`. It returns the resulting string.

```plaintext
procedure format(fmt, v[])
  v := copy(v)
  r := ""
  fmt ? {
    while r ||:= tab(upto('%')) do {
      move(1)
      r||:= (
        case move(1) of {
          "v":1
          "i":image
          "I":Image})(get(v))
      }
      r ||:= tab(0)
    }
    return r
  end
```

Problem 4: (8 points)

Write a program `vc` that reads lines on standard input and prints those lines that contain more vowels than consonants.

```plaintext
procedure main()
  while line := read() do {
    vc := cc := 0
    map(line) ? while c := move(1) do {
      case c of {
        !'aeiou': vc += 1
        !(&lcase--'aeiou'): cc += 1
      }
    }
    vc > cc & write(line)
  }
end
```
There were some interesting approaches to counting. Mr. Jeffrey and Ms. Yost came up with this:

\[
\text{every upto('aeiou') do v +:= 1}
\]

Mr. Graham did this:

\[
\text{tab(any(vowels)) & v +:= 1}
\]

**Problem 5: (25 points)**

Write a procedure \texttt{dollars(s)} that converts a string specifying a sum of money into a corresponding real value.

```
procedure dollars(s)
local dollars, cents
s ? { 
  value := {
    cents := tab(many(&digits)) & *cents <= 2 & ="c" & cents * .01 } |
    =$" & dollars := tab(many(&digits)) & ="." & cents := tab(many(&digits)) & *cents = 2 & dollars + cents * .01 } |
    =("one buck"|"one dollar") & 1.0 } |
    =(english(n := 2 to 20) || (" dollars"|" bucks")) & real(n) |
  } & pos(0) & return value
} & end
```

**Problem 6: (9 points)**

Write a PDCO \texttt{Longest{expr1, expr2, ..., exprN}} that returns the argument with the longest result sequence.

```
procedure Longest(L)
R := []
every c := !L do {
  while @c
    put(R, [c, *c])
}
return ^(sortf(R, 2)[-1][1])
end
```
Mr. Linn's solution was the easiest to understand:

```icon
procedure Longest(L)
    max := 0
    every d := !L do {
        while @d
            if *d > max then c := ^d
    }
    return \c
end
```

Problem 7: (2 points each; 6 points total)

(a) What is the fundamental difference between an additive color model and a subtractive color model?

In an additive color model light the component colors contribute energy to produce a resulting color.

In a subtractive model ink of the component colors absorbs light of various wavelengths. The light is not absorbed (and thus reflected) is what the viewer sees.

(b) Name a subtractive color model and the colors it uses.

The CMY (cyan, magenta, yellow) color model is subtractive.

(c) What "color" would be produced by the setting Fg("#bababa")?

It would be gray, which technically isn't a color.

**EXTRA CREDIT SECTION**

(a) (1 point) What is the shortest Icon program that will successfully compile and execute without error?

```icon
record main()
```

(b) (1 point) List the last names of ten other students in this class.

Mr. Thayer and Mr. Yee either learn from experience or are social butterflies.

(c) (3 points) An Icon programmer homesick for Java wants to produce output with System.out.println() instead of write(). Create a file java.icn that provides the necessary elements to meet the needs of this misdirected individual until professional help can be obtained.

```icon
record java_sys(out)
record java_out(println)
global System
procedure java_init()
    System := java_sys(java_out(write))
end
```