STATS 782 Assignment 4; University of Auckland, Semester 1

Due Date: 23:59 NZ Time, Thursday 2 Jun 2022

Chase Robertson, crob873

I have read the declaration on the cover sheet and confirm my agreement with it.

Question 1

a)

```
# pgon s3 class constructor
pgon <- function(x, y) {
    # check for valid x,y args
    if (length(x) < 1 || length(y) < 1)
        stop("empty argument vector")
    if (!all(is.numeric(c(x, y))) || !all(is.finite(c(x, y))))
        stop("invalid vector contents")

    # recycle x or y to same length if necessary
    suppressWarnings(xy <- cbind(x, y))

    # recycle both to min length 3 if necessary
    if (nrow(xy) < 3)
        xy <- rbind(xy, xy, xy)[1:3,]

    # create pgon object
    pg <- list(x=xy[,1], y=xy[,2])
    class(pg) <- "pgon"
    pg
}

# test pgon constructor
(p4 <- pgon(c(0,1,1), c(0,0,1,1)))
```

```
## $x
## [1] 0 1 1 0
##
## $y
## [1] 0 0 1 1
##
## attr(,"class")
## [1] "pgon"
```

```
a  <- seq(0, 2*pi, length.out=9)[1:8]
(p8 <- pgon(sin(a), cos(a)))
```

```
## $x
## [1]  0.000000e+00  7.071068e-01  1.000000e+00  7.071068e-01  1.224647e-16
```

```
## [6] -7.071068e-01 -1.000000e+00 -7.071068e-01
##
## $y
## [1]  1.000000e+00  7.071068e-01  6.123234e-17 -7.071068e-01 -1.000000e+00
## [6] -7.071068e-01 -1.836970e-16  7.071068e-01
##
## attr(,"class")
## [1] "pgon"
```

b)

```
# define S3 generic pts
pts <- function(x) UseMethod("pts")

# define pgon pts method, return x and y as matrix
pts.pgon <- function(pg) {
    xy <- cbind(pg$x, pg$y)
    colnames(xy) <- c("x", "y")
    xy
}

pts(p4)
```

```
##      x y
## [1,] 0 0
## [2,] 1 0
## [3,] 1 1
## [4,] 0 1
```

c)

```
# define pgon length as number of points
length.pgon <- function(pg) length(pg$x)

length(p8)
```

```
## [1] 8
```

d)

```
# pretty-print pgon
print.pgon <- function(pg) {
    n <- length(pg)
    # max number of rows to show
    n_shown <- 4

    cat("Polygon of", n, "points.", fill=T)

    print(head(pts(pg), n_shown))

    if (n > n_shown) cat("[...]", fill=T)
}

p8
```

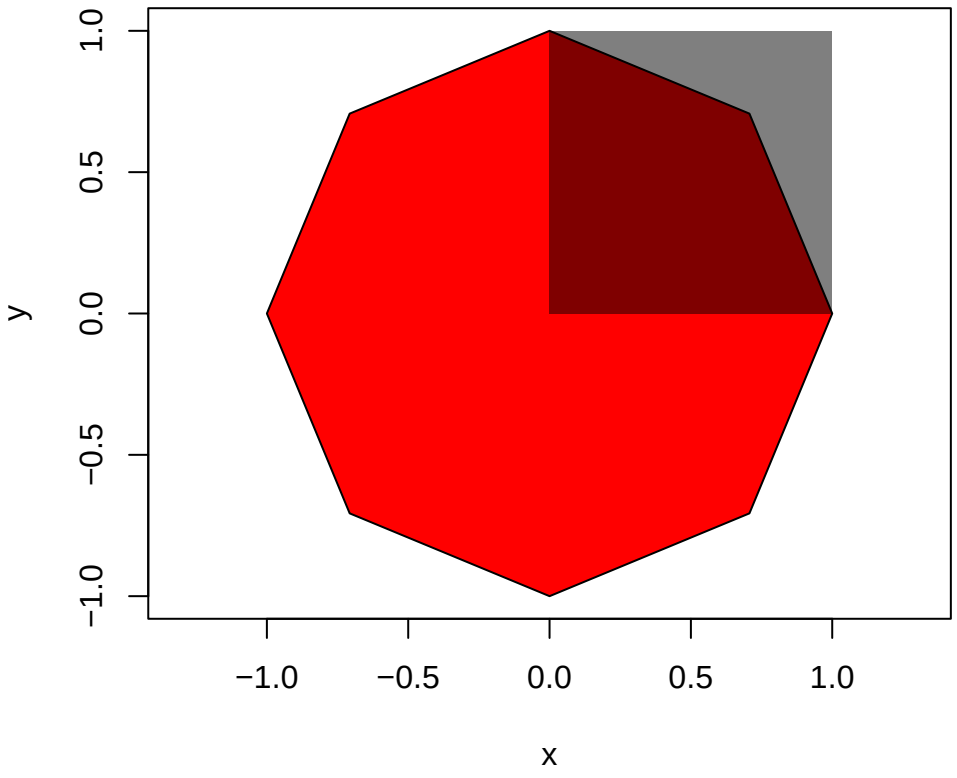
```
## Polygon of 8 points.
##              x             y
## [1,] 0.0000000  1.000000e+00
## [2,] 0.7071068  7.071068e-01
## [3,] 1.0000000  6.123234e-17
## [4,] 0.7071068 -7.071068e-01
## [...]
```

e)

```
# pgon plot method
plot.pgon <- function(pg, add=FALSE, asp=1, ...) {
    # if not adding, create an empty plot
    if (!add) plot(pts(pg), type='n', asp=asp)

    # add polygon to most recently created plot
    polygon(pts(pg), ...)
}

par(mar=c(4, 4, 0.1, 0.1))
plot(p8, col="red")
plot(p4, col="#00000080", border=NA, add=TRUE)
```



f)

```
check_pgon_op <- function(x)
    # ensure x is length 2 numeric vector
    if (length(x) != 2 || !all(is.numeric(x)))
        stop("invalid arg to pgon operation")
```

```
Ops.pgon <- function(e1, e2=NULL) {
    op <- get(.Generic)

    # unary operation
    if (is.null(e2)) {
        e1$x <- op(e1$x)
        e1$y <- op(e1$y)
        e1
    }

    # binary operation
    else {
        if (is(e1, "pgon")) {
            # if e1 is pgon, e2 should be numeric vector
            check_pgon_op(e2)
            x <- op(e1$x, e2[1])
            y <- op(e1$y, e2[2])
        }
        else {
            # e1 not pgon, so e1 should be numeric vector
            check_pgon_op(e1)
            x <- op(e1[1], e2$x)
            y <- op(e1[2], e2$y)
        }
        pgon(x, y)
    }
}

# test add/subtract pgons
-p4
```

```
## Polygon of 4 points.
##       x  y
## [1,]  0  0
## [2,] -1  0
## [3,] -1 -1
## [4,]  0 -1
```

```
p4 + c(1,2)
```

```
## Polygon of 4 points.
##      x y
## [1,] 1 2
## [2,] 2 2
## [3,] 2 3
## [4,] 1 3
```

```
p4 - pts(p8)[1,]
```

```
## Polygon of 4 points.
##      x  y
## [1,] 0 -1
## [2,] 1 -1
## [3,] 1  0
## [4,] 0  0
```

```
c(1, 1) - p4
```

```
## Polygon of 4 points.
##      x y
## [1,] 1 1
## [2,] 0 1
## [3,] 0 0
## [4,] 1 0
```

g)

```
# coloured polygon constructor
colygon <- function(x, y=NULL, col="grey") {
    # if one pgon arg, keep it, otherwise create new pgon
    cg <- if (is.null(y) && is(x, "pgon")) x else pgon(x, y)

    # add colygon attributes
    cg$col <- col
    class(cg) <- c("colygon", class(cg))
    cg
}

# pretty-print coloured polygon
print.colygon <- function(cg) {
    n <- length(cg)
    # max number of rows to show
    n_shown <- 4

    cat(paste0("Coloured polygon of ", n, " points and colour ", cg$col, "."), fill=T)

    print(head(pts(cg), n_shown))

    if (n > n_shown) cat("[...]", fill=T)
}

(l3 <- colygon(c(-1, 0, 1), c(-1, 1), col="yellow"))
```

```
## Coloured polygon of 3 points and colour yellow.
##       x  y
## [1,] -1 -1
## [2,]  0  1
## [3,]  1 -1
```

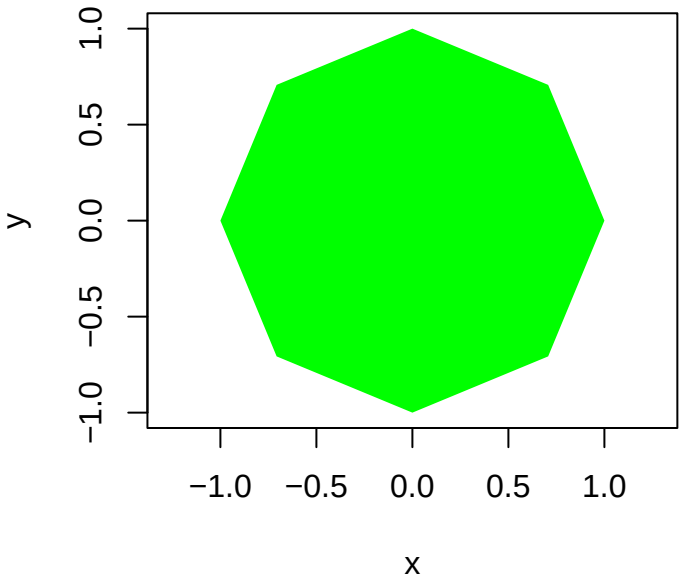
```
(l8 <- colygon(p8, col="green"))
```

```
## Coloured polygon of 8 points and colour green.
##              x             y
## [1,] 0.0000000  1.000000e+00
## [2,] 0.7071068  7.071068e-01
## [3,] 1.0000000  6.123234e-17
## [4,] 0.7071068 -7.071068e-01
## [...]
```

h)

```
plot.colygon <- function(cg, ...) {
    # use pgon method to plot colygon
    NextMethod(cg, col=cg$col, border=NA, ...)
}

# test colygon plot
plot(l8)
```



i)

```
l8 + c(0, 1)
```

```
## Polygon of 8 points.
##              x         y
## [1,] 0.0000000 2.0000000
## [2,] 0.7071068 1.7071068
## [3,] 1.0000000 1.0000000
## [4,] 0.7071068 0.2928932
## [...]
```

```
-l8
```

```
## Coloured polygon of 8 points and colour green.
##               x            y
## [1,]  0.0000000 -1.000000e+00
## [2,] -0.7071068 -7.071068e-01
## [3,] -1.0000000 -6.123234e-17
## [4,] -0.7071068  7.071068e-01
## [...]
```

Because the `Ops` group method has not been explicitly defined for class `colygon`, the next class `pgon` is used to resolve calls to the `Ops` method on a `colygon`. The `Ops.pgon` method's unary case simply updates the `x` and `y` attributes and returns the object, with class information and other attributes intact, so the `colygon` class is maintained. In the binary case, however, a new `pgon` object is constructed from the operation's results, so subclass identification and attributes are not present in the returned object.

```r
# define operations on coloured polygons
Ops.colygon <- function(e1, e2=NULL) {
    # use pgon operation behaviour
    gon <- NextMethod(e1, e2)

    # reconstruct colygon with e1 colour if necessary
    if (is(gon, "colygon")) gon
    else colygon(gon, col=e1$col)
}

l8 + c(0, 1)
```

```
## Coloured polygon of 8 points and colour green.
##              x         y
## [1,] 0.0000000 2.0000000
## [2,] 0.7071068 1.7071068
## [3,] 1.0000000 1.0000000
## [4,] 0.7071068 0.2928932
## [...]
```

```r
-l8
```

```
## Coloured polygon of 8 points and colour green.
##               x             y
## [1,]  0.0000000 -1.000000e+00
## [2,] -0.7071068 -7.071068e-01
## [3,] -1.0000000 -6.123234e-17
## [4,] -0.7071068  7.071068e-01
## [...]
```

Question 2

(No need to answer here, upload the final package to Canvas!)