# STATS 102A Homework 4

Winter 2022

Takao Oba
205615894

Due 02/27/2022

## Sourcing the Functions

```
source("205615894_stats102a_hw4.R")
```

## Setting the Precision

```
options(digits = 10)
```

Alternatively, you can use specify the `digits` argument inside `print` when printing long numbers.

# 1 Dealing with Large Numbers

## 1(a) Basic Functions

Write a constructor function, an appropriate predicate function, appropriate coercion functions, and a useful `print()` method. This question accounts for 25% of this assignment.

You are expected to provide the following functions (refer to `homework4.pdf` for function requirements):

- `pqnumber(sign, p, q, nums)`
- `is_pqnumber(x)`
- `print(x, DEC)`
- `as_pqnumber(x, p, q)`
- `as_numeric(x)`

Create three `pqnumber` objects for the following numbers to demonstrate each function:

1. sign = 1, p = 3, q = 4, nums = [1 2 3 4 5 6 7 8], and the decimal value = 87654.321
2. sign = 1, p = 6, q = 0, nums = [3 9 5 1 4 1 3], and the decimal value = 3141593
3. sign = -1, p = 5, q = 1, nums = [2 8 2 8 1 7 2], and the decimal value = -27.18282

## 1(b) Test Cases for pqnumber()

```
num1 <- pqnumber(1, 3, 4, 1:8)
num2 <- pqnumber(1, 6, 0, c(3,9,5,1,4,1,3))
num3 <- pqnumber(-1, 5, 1, c(2,8,2,8,1,7,2))
num1
```

```
## [1] sign =  1
## [1] p =  3
## [1] q =  4
## [1] nums =  1       2       3       4       5       6       7       8
```

```
num2
```

```
## [1] sign =  1
## [1] p =  6
## [1] q =  0
## [1] nums =  3       9       5       1       4       1       3
```

```
num3
```

```
## [1] sign =  -1
## [1] p =  5
## [1] q =  1
## [1] nums =  2       8       2       8       1       7       2
```

## 1(c) Test Cases for is_pqnumber()

```
is_pqnumber(num1)
```

```
## [1] TRUE
```

```
is_pqnumber(1)
```

```
## [1] FALSE
```

```
is_pqnumber("pqnumber")
```

```
## [1] FALSE
```

## 1(d) Test Cases for print()

```
print(num1, DEC=T)
```

```
## [1] sign =  1
## [1] p =  3
## [1] q =  4
## [1] nums =  1       2       3       4       5       6       7       8
## [1] Decimal =  87654.321
```

```
print(num1, DEC=F)
```

```
## [1] sign =  1
## [1] p =  3
## [1] q =  4
## [1] nums =  1       2       3       4       5       6       7       8
```

```
print(num2, DEC=T)
```

```
## [1] sign =  1
## [1] p =  6
## [1] q =  0
## [1] nums =  3       9       5       1       4       1       3
## [1] Decimal =  3.141593
```

```
print(num3, DEC=T)
```

```
## [1] sign =  -1
## [1] p =  5
## [1] q =  1
## [1] nums =  2       8       2       8       1       7       2
## [1] Decimal =  -27.18282
```

### 1(e) Test Cases for as_numeric()

```
as_numeric(num1)
```

```
## [1] 87654.321
```

```
as_numeric(num2)
```

```
## [1] 3.141593
```

```
as_numeric(num3)
```

```
## [1] -27.18282
```

### 1(f) Test Cases for as_pqnumber()

```
as_pqnumber(87654.321, 3, 5)
```

```
## [1] sign =  1
## [1] p =  3
## [1] q =  5
##  [1] nums =  1       2       3       4       5       6       7       8
## [10] 0
```

```
as_pqnumber(3, 3, 3)
```

```
## [1] 0
## [1] 0
```

```
## [1] sign =  1
## [1] p =  3
## [1] q =  3
## [1] nums =  0       0       0       3       0       0       0
```

```
as_pqnumber(0, 3, 0)
```

```
## [1] sign =  0
## [1] p =  3
## [1] q =  0
## [1] nums =  0       0       0       0
```

```
as_pqnumber(pi, 6, 1)
```

```
## [1] sign =  1
## [1] p =  6
## [1] q =  1
## [1] nums =  3       9       5       1       4       1       3       0
```

```
as_pqnumber(-pi, 5, 2)
```

```
## [1] sign =  -1
## [1] p =  5
## [1] q =  2
## [1] nums =  9       5       1       4       1       3       0       0
```

## 1(g) Addition and Subtraction

Write an addition function `add(x, y)` and a subtraction function `subtract(x, y)`. This question accounts for 60% of this assignment.

1. Define a carry-over function for adding two numbers, which moves the extra digits in the appropriate way.
2. Likewise, a subtraction function should have a borrowing function that borrows 10 in the same way as you would do a subtraction with pencil-and-paper.

3. Your functions should work for both positive and negative `pqnumber` objects. Both functions should return a `pqnumber` object with enough `p` and `q` to carry the result.

```
knitr::include_graphics("carry_over.png")
```

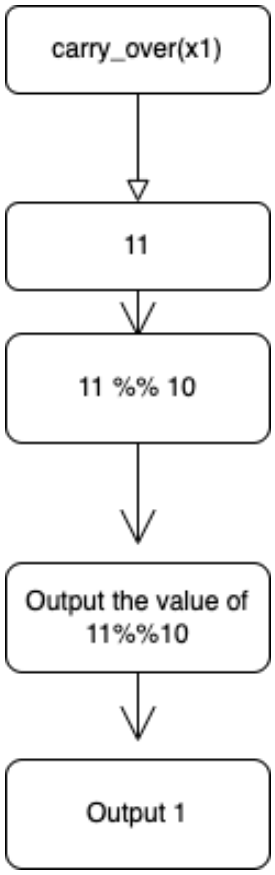


Figure 1: General stucture of a flowchart.

```
knitr::include_graphics("add.png")
```

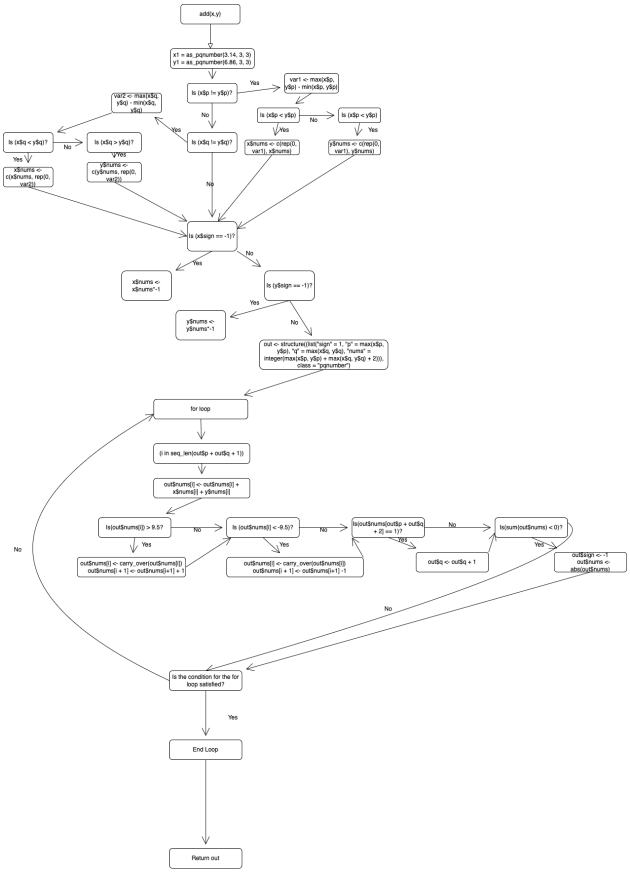


Figure 2: General stucture of a flowchart.

## 1(h) Test Cases for add()

```
x1 = as_pqnumber(3.14, 3, 3)
y1 = as_pqnumber(6.86, 3, 3)
print(add(x1, y1), DEC=T)
```

```
## [1] sign =  1
## [1] p =  3
## [1] q =  3
## [1] nums =  0       0       0       0       1       0       0       0
## [1] Decimal =  10
```

```
x11 = x1; x11$sign = -1
y11 = y1; y11$sign = -1
print(add(x1, y11), DEC=T)
```

```
## [1] sign =  -1
## [1] p =  3
## [1] q =  3
## [1] nums =  0       2       7       3       0       0       0       0
## [1] Decimal =  -3.72
```

```
print(add(x11, y11), DEC=T)
```

```
## [1] sign =  -1
## [1] p =  3
## [1] q =  3
## [1] nums =  0       0       0       0       1       0       0       0
## [1] Decimal =  -10
```

```
print(add(x11, y1), DEC=T)
```

```
## [1] sign =  1
## [1] p =  3
## [1] q =  3
## [1] nums =  0       2       7       3       0       0       0       0
## [1] Decimal =  3.72
```

```
print(add(num1, num2), DEC=T)
```

```
## [1] sign =  1
## [1] p =  6
## [1] q =  4
##  [1] nums =  3       9       5       2       6       4       7       5
## [10] 6       7       8       0
## [1] Decimal =    87657.462593
```

```
print(add(num1, num1), DEC=T)
```

```
## [1] sign =  1
## [1] p =  3
## [1] q =  5
##  [1] nums =  2       4       6       8       0       3       5       7
## [10] 1
## [1] Decimal =  175308.642
```

```
knitr::include_graphics("burrowing.png")
```

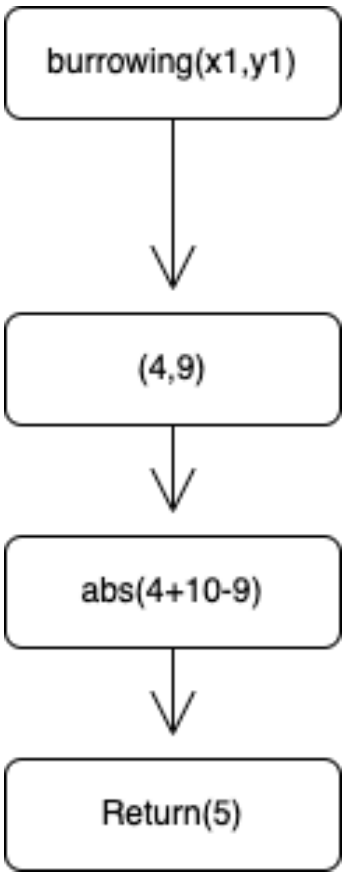


Figure 3: General stucture of a flowchart.

```
knitr::include_graphics("subtract.png")
```

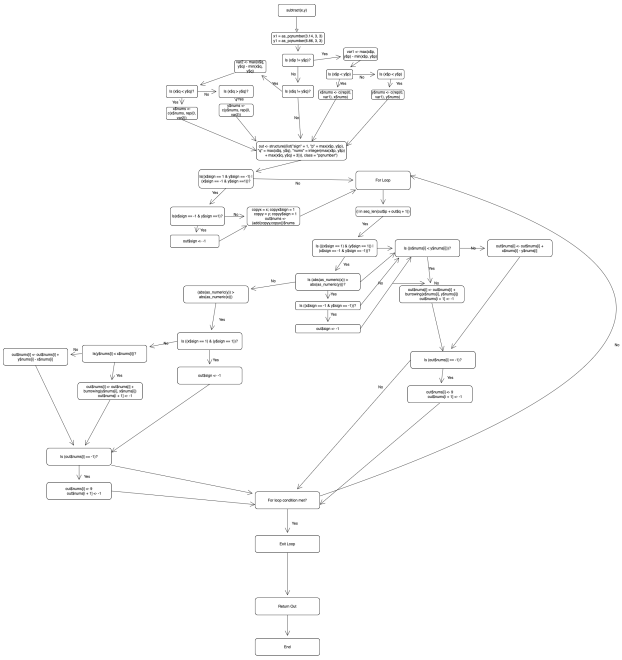

Figure 4: General stucture of a flowchart.

## 1(i) Test Cases for subtract()

```
x2 = as_pqnumber(3.14, 3, 3)
y2 = as_pqnumber(-6.86, 3, 3)
subtract(x2, y2)
```

```
## [1] sign =  1
## [1] p =  3
## [1] q =  3
## [1] nums =  0  0  0  0  1  0  0  0
```

```
subtract(y2, x2)
```

```
## [1] sign =  -1
## [1] p =  3
## [1] q =  3
## [1] nums =  0  0  0  0  1  0  0  0
```

```
print(subtract(num2, num1), DEC=T)
```

```
## [1] sign =  -1
## [1] p =  6
## [1] q =  4
##  [1] nums =  7  0  4  9  7  1  1  5
## [10] 6  7  8  0  0
## [1] Decimal =  -87651.179407
```

```
num11 = num1; num11$sign = -1
num22 = num2; num22$sign = -1
print(subtract(num11, num22), DEC=T)
```

```
## [1] sign =  -1
## [1] p =  6
## [1] q =  4
##  [1] nums =  7       0       4       9       7       1       1       5
## [10] 6       7       8       0       0
## [1] Decimal =     -87651.179407
```

```
x = as_pqnumber(654.321, 3, 5)
y = as_pqnumber(543.21, 3, 4)
subtract(y, x)
```

```
## [1] sign =  -1
## [1] p =  3
## [1] q =  5
##  [1] nums =  1       1       1       1       1       1       0       0
## [10] 0       0       0
```

```
subtract(x, y)
```

```
## [1] sign =  1
## [1] p =  3
## [1] q =  5
##  [1] nums =  1       1       1       1       1       1       0       0
## [10] 0       0       0
```

## 1(j) Multiplication

Use your `add()` function to write a multiplication function `multiply(x, y)` which can multiply two `pqnumber` objects `x` and `y`. Think about how you would multiply two large numbers by hand and implement that algorithm in R for two `pqnumber` objects. The function should also return a `pqnumber` object. Both functions should return a `pqnumber` object with enough `p` and `q` to carry the result. This question accounts for 15% of this assignment.

**Note**: Please attach the flowchart or algorithms for your *carry-over*, **addition**, **subtraction**, **borrowing**, and **multiplication** functions. Also, the cases provided here are only for your test. We will use different arguments and objects to try your functions while grading. Therefore, try your best to make your functions efficient, accurate, and robust.

```
knitr::include_graphics("multiply.png")
```

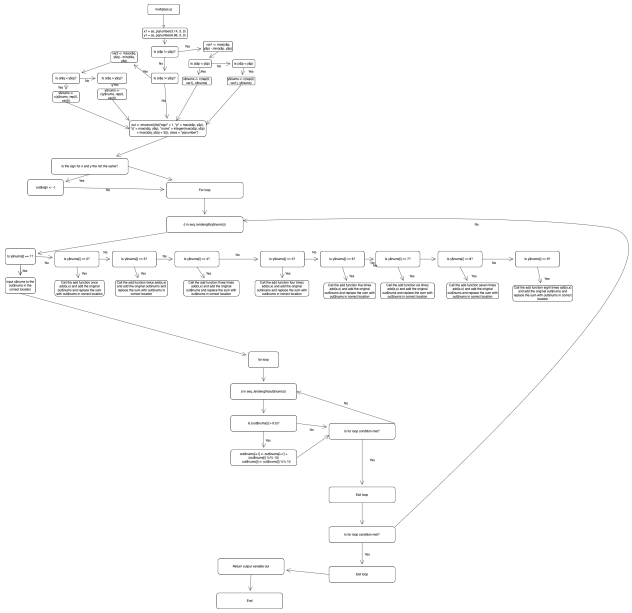

Figure 5: General stucture of a flowchart.

## 1(k) Test Cases for multiply()

```
x3 = as_pqnumber(654.321, 3, 5)
y3 = as_pqnumber(543.21, 3, 4)
print(multiply(x3, y3), DEC=T)
```

```
## [1] sign =  1
## [1] p =  6
## [1] q =  9
##  [1] nums =  0       1       4       0       1       7       3       3
## [10] 4       5       5       3       0       0       0       0
## [1] Decimal =    355433.71041
```

```
print(multiply(y3, x3), DEC=T)
```

```
## [1] sign =  1
## [1] p =  6
## [1] q =  9
##  [1] nums =  0       1       4       0       1       7       3       3
## [10] 4       5       5       3       0       0       0       0
## [1] Decimal =    355433.71041
```

```
x4 <- as_pqnumber(0, 2, 1)
y14 <- as_pqnumber(-5, 2, 1)
```

```
## [1] 0
```

```
print(multiply(x4, y14), DEC=T)
```

```
## [1] sign =  -1
## [1] p =  4
## [1] q =  2
## [1] nums =  0       0       0       0       0       0       0
## [1] Decimal =  0
```

```
x5 = as_pqnumber(12345.6, 3, 5)
```

```
## [1] 0
```

```
y5 = as_pqnumber(98765.43, 4, 5)
```

```
## [1] 0
## [1] 0
```

```
print(multiply(x5, y5), DEC = T)
```

```
## [1] sign =  1
## [1] p =  7
## [1] q =  10
##  [1] nums =  0       0       0       0       0       8       0       6
## [10] 2       9       4       8       1       3       9       1       2
## [19] 1
## [1] Decimal =      12193184926.08
```