

# hw\_1

Sonish Lamsal, Emmie Jenkins

1/27/2021

## Question 1(a)

```
divide <- function(d,a) {  
  if (a==0) stop("division by zero is undefined.")  
  if (d<0 || a<0) stop("divide() only works for positive 'a' and 'd'.")  
  quotient <- 0  
  while (a<=d){  
    d = d - a  
    quotient = quotient + 1  
  }  
  c(q = quotient, r = d)  
}  
divide(22, 7)
```

```
## q r  
## 3 1
```

```
22 %/% 7
```

```
## [1] 3
```

```
22 %% 7
```

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

```
divide(22, 0)
```

```
## Error in divide(22, 0): division by zero is undefined.
```

```
divide(-22,7)
```

```
## Error in divide(-22, 7): divide() only works for positive 'a' and 'd'.
```

```
divide(21,7)
```

```
## q r  
## 3 0
```

## Question 1(b)

```
mod <- function(d,a) {  
  a<- divide(d,a)  
  unname(a[2])  
}  
mod(23, 7)
```

```
## [1] 2
mod(21,7)
```

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

Question 1(c)

Question 1(d)

```
is.divisor <- function(d,a) {
  if (a==0){
    rem <- 1
  }
  else rem <- mod(abs(d),abs(a))
  rem == 0
}
is.divisor(6, 3)
```

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

```
is.divisor(6, 4)
```

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

```
is.divisor(-6, 3)
```

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

```
is.divisor(6, 0)
```

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

Question 1(e)

```
divisors <- function(d) {
  if (d==0) stop("Input cannot be zero")
  num <- c(-abs(d):abs(d))
  check <- lapply(num, is.divisor, d=d)
  num[unlist(check)]
}
divisors(7)
```

```
## [1] -7 -1  1  7
```

```
divisors(18)
```

```
## [1] -18 -9 -6 -3 -2 -1  1  2  3  6  9 18
```

Question 1(f)

```
gcd_naive <- function(a,b) {
  if ( a!=0 && b!=0){
    div_a <- divisors(a)
    div_b <- divisors(b)
    common <- intersect(div_a, div_b)
    gcd <- max(common)
  }
}
```

```

    else{
      gcd <- max(abs(a),abs(b))
    }
    gcd
  }
gcd_naive(64, 28)

```

```
## [1] 4
```

```
gcd_naive(64, -28)
```

```
## [1] 4
```

```
gcd_naive(64,0)
```

```
## [1] 64
```

```
gcd_naive(-64,0)
```

```
## [1] 64
```

```
gcd_naive(0,0)
```

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

### Question 1(g)

```

is.prime_naive <- function(p) {
  div <- divisors(p)
  length(div) == 4
}
is.prime_naive(5)

```

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

```
is.prime_naive(20)
```

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

```
is.prime_naive(1)
```

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

```
is.prime_naive(-3)
```

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

### Question 1(h)

```

naive_seive <- function(n) {
  num <- c(1:n)
  check <- lapply(num, is.prime_naive)
  num[unlist(check)]
}
naive_seive(100)

```

```
## [1] 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

### Question 1(i)

```
eratosthenes_sieve <- function(n) {  
  numbers <- c(2:n)  
  check <- rep(TRUE, n-1)  
  value <- floor(sqrt(n))  
  for (i in 2:value) {  
    mul <- n %/% i  
    for (j in 2:mul) {  
      check[match(i*j,numbers)] <- FALSE  
    }  
  }  
  numbers[check]  
}  
eratosthenes_sieve(100)
```

```
## [1] 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

### Question 1(j)

```
is.relprime <- function(a, b) {  
  gcd_naive(a, b) == 1  
}  
is.relprime(9, 16)
```

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

```
is.relprime(6, 21)
```

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

### Question 1(k)

```
gcd_recursive <- function(a, b, quiet=TRUE) {  
  if (quiet == FALSE) {  
    cat("(" ,a ,",",b ,")\n")  
  }  
  if (a==0 && b==0) 0  
  if (a==0 || b==0){  
    max(abs(a),abs(b))  
  }  
  else{  
    c <- abs(min(a,b))  
    d <- abs(max(a,b))  
    d <- mod(d,c)  
    gcd_recursive(c,d, quiet)  
  }  
}  
gcd_recursive( 64, 28, quiet = FALSE)
```

```
## ( 64 , 28 )  
## ( 28 , 8 )  
## ( 8 , 4 )  
## ( 4 , 0 )
```

```
## [1] 4
gcd_recursive( 64, -28)
```

```
## [1] 4
gcd_recursive( 64, 0)
```

```
## [1] 64
gcd_recursive(-64, 0)
```

```
## [1] 64
gcd_recursive( 0, 0)
```

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

### Question 1(l)

```
gcd <- function(a,b, quiet = TRUE) {
  if (quiet == FALSE) {
    cat("(" ,a ,",",b ,")\n")
  }
  if (a==0 && b==0) 0
  else if (a==0 || b==0) max(abs(a),abs(b))
  else {
    while (a!=0 && b!=0) {
      c <- abs(min(a,b))
      d <- abs(max(a,b))
      a <- c
      b <- mod(d,c)
      if (quiet == FALSE) {
        cat("(" ,a ,",",b ,")\n")
      }
    }
    max(abs(a),abs(b))
  }
}

gcd( 64, 28, quiet = FALSE)
```

```
## ( 64 , 28 )
## ( 28 , 8 )
## ( 8 , 4 )
## ( 4 , 0 )
```

```
## [1] 4
gcd( 64, -28)
```

```
## [1] 4
gcd( 64, 0)
```

```
## [1] 64
gcd(-64, 0)
```

```
## [1] 64
gcd( 0, 0)
```

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

## Question 1(m)

```
library(bench)
(result <- mark(
  gcd(64,28),
  gcd_recursive(64,28),
  relative = TRUE
))
```

```
## # A tibble: 2 x 6
##   expression          min median `itr/sec` mem_alloc `gc/sec`
##   <bch:expr>      <dbl>  <dbl>    <dbl>    <dbl>    <dbl>
## 1 gcd(64, 28)         1      1      1.04      NaN     4.84
## 2 gcd_recursive(64, 28) 1.06   1.09      1      NaN      1
```

## Question 2(a)

```
inv <- function(b, x0 = 10^-(ceiling(log10(abs(b))))), tol = sqrt(.Machine$double.eps),
  message = FALSE) {
  if (b==0) stop("zero has no multiplicative inverse")
  dif <- 1 #initilaizing a difference > tol
  if (b > 0){ #for positive input b
    while (dif > tol){
      y <- x0 #setting the intial x iteration to y
      x0 <- x0 * (2 - b * x0)
      dif <- x0 - y #subtracting the newly iteration from the previous iteration
      if (message) print(x0) #for message=TRUE
    }
    x0
  }
  else { #for negative input b
    b <- -b
    while (dif > tol){
      y <- x0
      x0 <- x0 * (2 - b * x0)
      dif <- x0 - y
    }
    -x0
  }
}

inv(1000, message = TRUE)
```

```
## [1] 0.001
```

```
## [1] 0.001
```

```

inv(-1000)

## [1] -0.001
inv(0)

## Error in inv(0): zero has no multiplicative inverse
divide_real_fast <- function(a, b) {
  if (b==0) stop("division by zero is undefined.")
  a * inv(b)
}

divide_real_fast(22, 7)

## [1] 3.142857
divide_real_fast(2, -4)

## [1] -0.5
divide_real_fast(2, 0)

## Error in divide_real_fast(2, 0): division by zero is undefined.
divide_fast <- function(a, d){
  divide(a,d)
  c <- zapsmall(divide_real_fast(a,d))
  floor_c <- floor(c)
  t <- c - floor_c
  q <- floor_c
  r <- round(d * t) #computing remainder and rounding
  c(q = q, r = r)
}

a <- 1e8; d <- 2
divide(a, d)

##      q      r
## 5e+07 0e+00
divide_fast(a, d)

##      q      r
## 5e+07 0e+00
a %/% d

## [1] 5e+07
a %% d

## [1] 0

```