Assignment 2

Student Name : 정석규

Student ID : 201724570

Q1.

1) Source code :

```
region = data.frame(latitude = c(35.1796, 40.7128, 47.3769, 48.8566), longitude = c(129.0756, 74.0060, 8.5417, 2.3522), row.names = c('Busan', 'NewYork', 'Zurich', 'Paris'))

distance <- function(region) {



for (i in seq(dim(region)[1])) {
        if (row.names(region)[i] == 'NewYork') {
                region[i, 'longitude'] = -region[i, 'longitude']
        }
}

region = region*pi/180

comb_region <- combn(row.names(region), 2)

for (j in seq(dim(comb_region)[2])) {
        dis_region = region[comb_region[,j], ]
        pi_1 = dis_region[1,'latitude']; pi_2 = dis_region[2, 'latitude']; lambda_1 = dis_region[1, 'longitude']; lambda_2 = dis_region[2, 'longitude']
        central_angle = acos((sin(pi_1)*sin(pi_2))+(cos(pi_1)*cos(pi_2)*cos(abs(lambda_1-lambda_2))))
        Great_circle_distance = central_angle*6371
          cat("Distance between ", row.names(dis_region)[1]  ," and ", row.names(dis_region)[2], " is ", Great_circle_distance, "₩n")
        }
}
```

2) R Screenshot :

```
+ region = region*pi/180
+ comb_region <- combn(row.names(region), 2)
+ for (j in seq(dim(comb_region)[2])) {
+     dis_region = region[comb_region[,j], ]
+     pi_1 = dis_region[1,'latitude']; pi_2 = dis_region[2, 'latitude']; lambda_1 = dis_region[1, 'longitude']; lambda_2 = dis_region[2, 'longitude']
+     central_angle = acos((sin(pi_1)*sin(pi_2))+(cos(pi_1)*cos(pi_2)*cos(abs(lambda_1-lambda_2))))
+     Great_circle_distance = central_angle*6371
+     cat("Distance between ", row.names(dis_region)[1]  ," and ", row.names(dis_region)[2], " is ", Great_circle_distance, "\n")
+ }
+ }
> distance(region)
Distance between  Busan  and  NewYork  is  11252.36
Distance between  Busan  and  Zurich  is  9095.026
Distance between  Busan  and  Paris  is  9290.505
Distance between  NewYork  and  Zurich  is  6323.748
Distance between  NewYork  and  Paris  is  5837.241
Distance between  Zurich  and  Paris  is  487.8773
```

3) Answer : 1 – About 11252.36 km, 2 – Distance between "Busan" and "NewYork" is max distance as 11252.36 km among pair of cities.

Q2.

1) Source code :

```
value = 4834/200

val_to_con <- function(value) {



a = list()

while (TRUE) {

        a = append(a, floor(value))

        value = value - floor(value)

        value = 1/ value

        if (round(value, 4) == as.integer(value)) {

                a = append(a, as.integer(value))

                break;

        }

}

cont_frac = t(as.matrix(a))

return(cont_frac)

}

val_to_con(value)
```

```
con_frac <- c(3, 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1)

con_to_val <- function(con_frac) {


j = length(con_frac)

con_init = con_frac[1]


while ( j   !=   0 ) {

             con_init = 1/ con_init

             con_init = con_frac[j] + con_init

             j = j -1

}

return(con_init)

}

con_to_val(con_frac)
```

2) R Screenshot :

```
> value = 4834/200
> val_to_con <- function(value) {
+
+     a = list()
+     while (TRUE) {
+         a = append(a, floor(value))
+         value = value - floor(value)
+         value = 1/ value
+         if (round(value, 4) == as.integer(value)) {
+             a = append(a, as.integer(value))
+             break;
+         }
+     }
+     cont_frac = t(as.matrix(a))
+     return(cont_frac)
+ }
> val_to_con(value)
     [,1] [,2] [,3] [,4] [,5]
[1,] 24    5    1    7    2
```

```
> con_frac <- c(3, 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1)
> con_to_val <- function(con_frac) {
+
+     j = length(con_frac)
+     con_init = con_frac[1]
+
+     while ( j  !=  0 ) {
+         con_init = 1/ con_init
+         con_init = con_frac[j] + con_init
+         j = j -1
+     }
+     return(con_init)
+ }
> con_to_val(con_frac)
[1] 3.141593
```

3) Answer : 1 – [24; 5, 1, 7, 2], 2 – 3.141593(about pi)

Q3.

1) Source code :

```
theta <- seq(1:180)


Rotation <- function(theta) {


theta_rt <- theta*pi/180

u <- matrix(c(1,0), 2, 1)

v <- matrix(c(4,0), 2, 1)

result_matrix<- matrix(1, 1, length(theta_rt))


for (i in seq(length(theta_rt))) {

        Theta <- theta_rt[i]

        R<- matrix(c(cos(Theta), -sin(Theta), sin(Theta), cos(Theta)),2,2)

        u_hat <- R%*%u

        inner_abs <- t(u_hat)%*%v

        result_matrix[1,i] <- abs(inner_abs)

}

return(theta[which.min(result_matrix)])
```

```
}

Rotation(theta)
```

2) **R Screenshot :**

```
> theta <- seq(1:180)
> Rotation <- function(theta) {
+
+     theta_rt <- theta*pi/180
+     u <- matrix(c(1,0), 2, 1)
+     v <- matrix(c(4,0), 2, 1)
+     result_matrix<- matrix(1, 1, length(theta_rt))
+
+     for (i in seq(length(theta_rt))) {
+         Theta <- theta_rt[i]
+         R<- matrix(c(cos(Theta), -sin(Theta), sin(Theta), cos(Theta)),2,2)
+         u_hat <- R%*%u
+         inner_abs <- t(u_hat)%*%v
+         result_matrix[1,i] <- abs(inner_abs)
+     }
+     return(theta[which.min(result_matrix)])
+ }
> Rotation(theta)
[1] 90
```

3) **Answer :** degree of 90

Q4.

1) **Source code :**

```
v1 <- c(-1, -0.5, -0.2, 0.2, 0.5, 1)

v2 <- c(-2, 5, 0, 2, -3, -2)

v3 <- c(0, 2, 1, 3, -2, -1)


gram_sch <- function(v1, v2, v3) {

v <- matrix(c(v1, v2, v3), 6, 3)

u <- matrix(1, dim(v)[1], dim(v)[2])


for (i in ((dim(v)[2]-1) : dim(v)[2])) {

        u[ , 1] = v[ , 1]

        proj <- matrix(0, dim(v)[1], i-1)
```

```
        for (j in seq(i-1)) {
                proj[, j] = matrix((t(u[ ,j])%*%v[ ,i])/(t(u[ , j])%*%u[ , j])*u[ , j])
                if ( i == 2) {
                        u[ ,i] = v[ ,i] - proj
                }
        }
        if ( i > 2 ) {
                u[ ,i] = v[ , i] - rowSums(proj)
        }
}
return(u)
}
result <- gram_sch(v1, v2, v3)
cor(result)
```

2) R Screenshot :

```
> v1 <- c(-1, -0.5, -0.2, 0.2, 0.5, 1)
> v2 <- c(-2, 5, 0, 2, -3, -2)
> v3 <- c(0, 2, 1, 3, -2, -1)
>
> gram_sch <- function(v1, v2, v3) {
+     v <- matrix(c(v1, v2, v3), 6, 3)
+     u <- matrix(1, dim(v)[1], dim(v)[2])
+
+     for (i in ((dim(v)[2]-1) : dim(v)[2])) {
+         u[ , 1] = v[ , 1]
+         proj <- matrix(0, dim(v)[1], i-1)
+         for (j in seq(i-1)) {
+             proj[, j] = matrix((t(u[ ,j])%*%v[ ,i])/(t(u[ , j])%*%u[ , j])*u[ , j])
+             if ( i == 2) {
+                 u[ ,i] = v[ ,i] - proj
+             }
+         }
+         if ( i > 2 ) {
+             u[ ,i] = v[ , i] - rowSums(proj)
+         }
+     }
+     return(u)
+ }
> result <- gram_sch(v1, v2, v3)
```

```
> result
     [,1]       [,2]       [,3]
[1,] -1.0 -3.3953488  0.6802876
[2,] -0.5  4.3023256 -0.6428301
[3,] -0.2 -0.2790698  0.9371926
[4,]  0.2  2.2790698  2.0684828
[5,]  0.5 -2.3023256 -0.3514945
[6,]  1.0 -0.6046512  0.3083617
> cor(result)
              [,1]          [,2]          [,3]
[1,]  1.000000e+00  1.676598e-18 -1.486196e-16
[2,]  1.676598e-18  1.000000e+00 -2.338358e-16
[3,] -1.486196e-16 -2.338358e-16  1.000000e+00
```

3) **Answer :** 1 - orthogonal vectors by row, 2 – correlation among u1,u2,u3

```
> result
     [,1]       [,2]       [,3]
[1,] -1.0 -3.3953488  0.6802876
[2,] -0.5  4.3023256 -0.6428301
[3,] -0.2 -0.2790698  0.9371926
[4,]  0.2  2.2790698  2.0684828
[5,]  0.5 -2.3023256 -0.3514945
[6,]  1.0 -0.6046512  0.3083617
> cor(result)
              [,1]          [,2]          [,3]
[1,]  1.000000e+00  1.676598e-18 -1.486196e-16
[2,]  1.676598e-18  1.000000e+00 -2.338358e-16
[3,] -1.486196e-16 -2.338358e-16  1.000000e+00
```

Q5.

1) **Source code :**

```
f <- function(x) x^4 - 9*x^3 - 334*x^2 + 4416*x - 10080

f.prime <- function(x) 4*x^3 - 27*x^2 - 668*x + 4416


Newton <- function(x, tol=1e-10) {

while(abs(f(x)) > tol) {

        x <- x - (f(x) / f.prime(x))

        }

return(x)

}
```

```
Descent <- function(x, tol = 1e-10) {

alpha = 0.0001



while(abs(alpha*f.prime(x)) > tol) {

            x <- x - (alpha*f.prime(x))

            }

return(x)

}
```

2) R Screenshot :

```
> f <- function(x) x^4 - 9*x^3 - 334*x^2 + 4416*x - 10080
> f.prime <- function(x) 4*x^3 - 27*x^2 - 668*x + 4416
>
> Newton <- function(x, tol=1e-10) {
+     while(abs(f(x)) > tol) {
+         x <- x - (f(x) / f.prime(x))
+     }
+     return(x)
+ }
> Newton(-100)
[1] -20
> x0 <- Newton(-20)
> f(x0)
[1] 0
> Newton(0)
[1] 3
> x0 <- Newton(3)
> f(x0)
[1] 0
> Newton(100)
[1] 14
> x0 <- Newton(14)
> f(x0)
[1] 0
```

```
> Descent <- function(x, tol = 1e-10) {
+     alpha = 0.0001
+
+     while(abs(alpha*f.prime(x)) > tol) {
+         x <- x - (alpha*f.prime(x))
+     }
+     return(x)
+ }
> Descent(-20)
[1] -12.87693
> x0 <- Descent(-20)
> f(x0)
[1] -75615.39
> Descent(-10)
[1] -12.87693
> x0 <- Descent(-10)
> f(x0)
[1] -75615.39
> Descent(10)
[1] 13.06453
> x0 <- Descent(10)
> f(x0)
[1] -331.3932
> Descent(20)
[1] 13.06453
> x0 <- Descent(20)
> f(x0)
[1] -331.3932
```

3) **Answer :** 1 – [-20, 3, 14], 2- [-12.87, 13.06]