```
getAdjMatrix <- function(fname)
{
    library(igraph)
# 1) Read the file into a graph object
    graph <- read.graph(fname,format="edgelist")
# 2) count the vertices and egdes and print them
    print(vcount(graph))
    print(ecount(graph))
# 3) convert the graph object into an adjacency matrix
    mat <- get.adjacency(graph,sparse=TRUE)
          return(t(mat))
}

getTranstitionMatrix <- function(A)
{
library(Matrix)
# 1) column sum for input matrix
    column_sum <- colSums(A)
# 2) z matrix initialized to colSums(A).
    N <- nrow(A)
    z <- matrix(column_sum,nrow=1,ncol=N)
# 3) z changes with the colSums value filtered against '>0' condition
    zinitial <- 1/N
    z <- ifelse(z>0,0.15*zinitial,zinitial)
    z <- matrix(z,nrow=1,ncol=N)
# 4) summary sparse matrix created
    summarymatrix <- summary(A)
# Ones in summary matrix divided by colSum[j]
    summarymatrix[,3] = 1/column_sum[summarymatrix[,2]]
    dim <- c(N, N)
# 5) matrix regenerated from modified summary matrix
    matrix <- sparseMatrix(i=summarymatrix[,1] ,j=summarymatrix[,2],x=summarymatrix[,3], dims=dim)

    return(list(matrix,z))
 }

myPageRank <- function(T, z, niter)
{
N <- nrow(T)
# 1) Create an initial Nx1 PageRank vector called “xold”.
    xold <- matrix(1/N,nrow=N,ncol=1)
    xnew <- xold
# 2) For niter iterations calculate page rank
    for(iter in 1:niter)
    {    xnew <- (0.85 *(T %*% xold) )+matrix((z %*% xold),nrow=N,ncol=1)
    xold <- xnew    }
    return(xnew)
}
```