```
# Matrix inversion is usually a costly computation and there may be some benefit
# to caching the inverse of a matrix rather than compute it repeatedly. The
# following two functions are used to cache the inverse of a matrix.

# makeCacheMatrix creates a list containing a function to
# 1. set the value of the matrix
# 2. get the value of the matrix
# 3. set the value of inverse of the matrix
# 4. get the value of inverse of the matrixn

makeCacheMatrix <- function(x = matrix()) {
  inv <- NULL
  set <- function(y) {
    x <<- y
    inv <<- NULL
  }
  get <- function() x
  setinverse <- function(inverse) inv <<- inverse
  getinverse <- function() inv
  list(set=set, get=get, setinverse=setinverse, getinverse=getinverse)
}


# The following function returns the inverse of the matrix. It first checks if
# the inverse has already been computed. If so, it gets the result and skips the
# computation. If not, it computes the inverse, sets the value in the cache via
# setinverse function.

# This function assumes that the matrix is always invertible.

cacheSolve <- function(x, ...) {
  inv <- x$getinverse()
  if(!is.null(inv)) {
    message("getting cached data.")
    return(inv)
  }
  data <- x$get()
  inv <- solve(data)
  x$setinverse(inv)
  inv
}
```