```
######################################
#### SUGGESTED EXERCISE SOLUTIONS ####
######################################

##########
## 11.1 ##
##########
#(a)
myfib4 <- function(thresh,printme){
  if(printme){
    fib.a <- 1
    fib.b <- 1
    cat(fib.a,", ",fib.b,", ",sep="")
    repeat{
      temp <- fib.a+fib.b
      fib.a <- fib.b
      fib.b <- temp
      cat(fib.b,", ",sep="")
      if(fib.b>thresh){
        cat("BREAK NOW...")
        break
      }
    }
  } else {
    fibseq <- c(1,1)
    counter <- 2
    repeat{
      fibseq <- c(fibseq,fibseq[counter-1]+fibseq[counter])
      counter <- counter+1
      if(fibseq[counter]>thresh){
        break
      }
    }
    return(fibseq)
  }
}
myfib4(thresh=150,printme=TRUE)
myfib4(1000000,T)
myfib4(150,FALSE)
myfib4(1000000,printme=F)
```

```r
#(b)
##(i)
myfac <- function(int){
  result <- 1
  while(int>1){
    result <- result*int
    int <- int-1
  }
  return(result)
}
myfac(5)
myfac(12)
myfac(0)
##(ii)
myfac2 <- function(int){
  if(int<0){
    return(NaN)
  }
  result <- 1
  while(int>1){
    result <- result*int
    int <- int-1
  }
  return(result)
}
myfac2(5)
myfac2(12)
myfac2(0)
myfac2(-6)
```

```
##########
## 11.2 ##
##########
#(a)
comp <- function(P,i,t=12,y,plotit=TRUE,...){
  yseq <- 1:y
  values <- P*(1+i/(100*t))^(t*yseq)

  if(plotit){
    plot(yseq,values,type="s",...)
  } else {
    return(values)
  }
}
##(i)
comp(5000,4.4,y=10,plotit=F)[10]
##(ii)
comp(100,22.9,12,20,plotit=T,main="Compound interest
calculator",ylab="Balance (F)",xlab="Year (y)")
##(iii)
ann <- comp(100,22.9,1,20,plotit=F)
lines(1:20,ann,lty=2,type="s")
legend("topleft",lty=c(1,2),legend=c("monthly interest","annual
interest"))
#(b)
quad <- function(k1,k2,k3){
  if(any(c(missing(k1),missing(k2),missing(k3)))){
    return("At least one of k1, k2, k3 was missing")
  }
  x <- k2^2-4*k1*k3
  if(x<0){
    cat("No real roots\n")
  } else if(x==0){
    return(-k2/(2*k1))
  } else {
    return(c((-k2-x^0.5)/(2*k1),(-k2+x^0.5)/(2*k1)))
  }
}
##(i)
quad(k1=2,k2=-1,k3=-5)
quad(1,1,1)
##(ii)
quad(k1=1.3,k2=-8,k3=-3.13)
quad(2.25,-3,1)
quad(1.4,-2.2,-5.1)
quad(-5,10.11,-9.9)
##(iii)
quad(0)
```

```
##########
## 11.3 ##
##########
#(a)
foo <- list("a",c("b","c","d","e"),"f",c("g","h","i"))
lapply(foo,function(x) paste(x,"!",sep=""))
#(b)
facrec <- function(x){
  if(x==0){
    return(1)
  } else {
    return(x*facrec(x-1))
  }
}
##(i)
facrec(5)
##(ii)
facrec(12)
##(iii)
facrec(0)
#(c)
geolist <- function(x){
  geo <- function(nums){
    return(prod(nums)^(1/length(nums)))
  }

  for(i in 1:length(x)){
    if(!is.matrix(x[[i]])){
      x[[i]] <- geo(x[[i]])
    } else {
      x[[i]] <- apply(x[[i]],1,geo)
    }
  }
  return(x)
}
##(i)
foo <-
list(1:3,matrix(c(3.3,3.2,2.8,2.1,4.6,4.5,3.1,9.4),4,2),matrix(c(3.3,3.2,
2.8,2.1,4.6,4.5,3.1,9.4),2,4))
geolist(foo)
##(ii)
bar <- list(1:9,matrix(1:9,1,9),matrix(1:9,9,1),matrix(1:9,3,3))
geolist(bar)
```

```
##########
## 12.1 ##
##########
#(a)
facrec2 <- function(x){
  if(x<0){
    stop("'x' must be a positive integer")
  }

  if(x==0){
    return(1)
  } else {
    return(x*facrec2(x-1))
  }
}
##(i)
facrec2(5)
##(ii)
facrec2(8)
##(iii)
facrec2(-8)
#(b)
matinv <- function(x,noninv=NA,nonmat="not a matrix",silent=TRUE){
  if(!is.list(x)){
    stop("'x' must be a list")
  }

  n <- length(x)
  if(n==0){
    stop("'x' appears to be empty")
  }

  if(!is.character(nonmat)){
    warning("attempting to coerce 'nonmat' to a character string")
    nonmat <- as.character(nonmat)
  }

  for(i in 1:n){
    if(is.matrix(x[[i]])){
      attempt <- try(solve(x[[i]]),silent=silent)
      if(class(attempt)=="try-error"){
        x[[i]] <- noninv
      } else {
        x[[i]] <- attempt
      }
    } else {
      x[[i]] <- nonmat
    }
  }

  return(x)
}
```

```
##(i)
x <- list(1:4,matrix(1:4,1,4),matrix(1:4,4,1),matrix(1:4,2,2))
matinv(x)
##(ii)
matinv(x,noninv=Inf,nonmat=666)
##(iii)
matinv(x,noninv=Inf,nonmat=666,silent=F)
##(iv)
x <-
list(diag(9),matrix(c(0.2,0.4,0.2,0.1,0.1,0.2),3,3),rbind(c(5,5,1,2),c(2,
2,1,8),c(6,1,5,5),c(1,0,2,0)),matrix(1:6,2,3),cbind(c(3,5),c(6,5)),as.vec
tor(diag(2)))
matinv(x,noninv="unsuitable matrix")
##(v)
x <- "hello"
matinv(x)
##(vi)
x <- list()
matinv(x)
```

```
##########
## 12.2 ##
##########
#(a)
prog_test_fancy <- function(n,...){
  result <- 0
  progbar <- txtProgressBar(min=0,max=n,...)
  for(i in 1:n){
    result <- result + 1
    Sys.sleep(0.5)
    setTxtProgressBar(progbar,value=i)
  }
  close(progbar)
  return(result)
}
ence <- Sys.time()
prog_test_fancy(50,style=3,char="r")
differ <- Sys.time()
differ-ence
#(b)
myfibrec2 <- function(n){
  if(n<0){
    warning("Assuming you meant 'n' to be positive -- doing that 
instead")
    n <- n*-1
  } else if(n==0){
    stop("'n' is uninterpretable at 0")
  }

  if(n==1||n==2){
    return(1)
  } else {
    return(myfibrec2(n-1)+myfibrec2(n-2))
  }
}
myfibvectorTRY2 <- function(nvec){
  nterms <- length(nvec)
  result <- rep(0,nterms)
  progbar <- txtProgressBar(min=0,max=nterms,style=3,char="-")
  for(i in 1:nterms){
    attempt <- try(myfibrec2(nvec[i]),silent=T)
    if(class(attempt)=="try-error"){
      result[i] <- NA
    } else {
      result[i] <- attempt
    }
    setTxtProgressBar(progbar,value=i)
  }
  close(progbar)
  return(result)
}
##(i)
myfibvectorTRY2(nvec=c(3,2,7,0,9,13))
##(ii)
t1 <- Sys.time()
myfibvectorTRY2(1:35)
t2 <- Sys.time()
t2-t1
```

```
### This takes almost 1 minute on my machine... execution slows down as
the recursion gets deeper... recursion perhaps not so good for computing
Fibonacci sequence
#(c)
t1 <- Sys.time()
fibvec <- c(1,1,rep(NA,33))
for(i in 3:35){
  fibvec[i] <- fibvec[i-2]+fibvec[i-1]
}
fibvec
t2 <- Sys.time()
t2-t1
### This is substantially quicker than recursion!
```