```
#Problem 1
#c)
ts <- arima.sim(model = list (ar = c(0.6,-0.4)), n = 2000)
plot.ts(ts, main = "Time series length 2000")

ac <- acf(ts, type = "correlation", plot = T)
ac

#d)
pac <- acf(ts, type = "partial", plot = T)
pac


#Problem 3
#a)
#library("forecast")

hawaii<-read.table('hawaii-new.dat',col.names=c('year_month', 'total',
'west', 'east'))
ts_total <- ts(hawaii$total, start = 1970, frequency = 12)
ts_total
n <- 312

ym.ascycle = as.factor(rep(c(1,1,1,2,2,2,3,3,3,4,4,4),n/12))
ym.ascycle

total_lmod <- lm (log(total) ~ year_month + I(year_month^2) + ym.ascycle,
data = hawaii)
total_lmod

ts_log <- ts (total_lmod$fitted.values, start = 1970, freq = 12)

plot.ts(log(ts_total), col = 'blue' , ylab ='total tourists (log)',main ='log
of total tourists Hawaii')
lines(ts_log,col = 'red')
legend('topleft', col = c('blue', 'red'), lty = 1, legend =c('log
total','fitted log total'))

plot.ts(log(ts_total)-ts_log)
abline(h = 0, col ='blue')

#b)
log_total = log(hawaii$total)

p_1996_m <- data.frame(year_month = 9601:9612, ym.ascycle =
as.factor(c(1,1,1,2,2,2,3,3,3,4,4,4)))
p_1996_m

p_1996 <- predict(total_lmod, p_1996_m, interval = "confidence")
p_1996

p_1996.df <- as.data.frame(p_1996)
p_1996.df

p_1996_t <- ts(c(log_total[277:312], p_1996.df$fit), start = 1993, frequency
= 12)
p_1996_t
```

```
plot(p_1996_t, type="o")
lines(seq(1996,1997,by=1/12)[-13], p_1996.df$fit, col='red', lwd=3)

#c)
dec_ts_total<- decompose(ts_total)
plot(dec_ts_total)

stl_ts_total <- stl(ts_total, s.window=12)
plot(stl_ts_total)


#Problem4
#a)
yt <-scan('lt.txt')

st <- array(500)
st[1] <- 0.2
st

sigma <- array(500)
sigma[1] <-  2.26
sigma

Vt <- array(500)
vt <- array(500)
kt <- array(500)

Mu0 = 0.2; S02 = 2.25

#Kalman filter
for (i in 1:500)
{
  vt[i] = yt[i] - st[i]
  Vt[i] = sigma[i] + 0.25
  kt[i] = sigma[i]/Vt[i]
  st[i+1] = st[i] + kt[i]*vt[i]
  sigma[i+1] = (1-kt[i])*sigma[i] + 0.01
}

e_log_likelihood <- (-1/2) * length(vt) * log(2*pi) - sum(log(Vt) +
(vt^2)/Vt)/2
e_log_likelihood


#b)
yt_miss = yt
yt_miss[5]=0
yt_miss[100]=0
yt_miss[165]=0

st <- array(500)
st[1] <- 0.2

sigma <- array(500)
sigma[1] <-  2.26

Vt <- array(500)
vt <- array(500)
kt <- array(500)
```

```
#Kalman filter
for (i in 1:500)
{
  vt[i] = yt_miss[i] - st[i]
  Vt[i] = sigma[i] + 0.25
  kt[i] = sigma[i]/Vt[i]
  st[i+1] = st[i] + kt[i]*vt[i]
  sigma[i+1] = (1-kt[i])*sigma[i] + 0.01
}

e_log_likelihood_miss <- (-1/2) * length(vt-3) * log(2*pi) - sum(log(Vt) +
(vt^2)/Vt)/2
e_log_likelihood_miss


#c)
library(dlm)
#MLE
lt=function(x)
{
  ltm=dlm(FF = 1,
          V = x[1],
          GG = 1,
          W = x[2],
          m0 = 0,
          C0 = 10^7)
  return(ltm)
}
m.lt=dlmMLE(y = yt,
            parm = c(0.2,2.25),
            build = lt,
            lower=c(0,0),
            upper=c(100,100),
            hessian = TRUE,
            control = list(maxit = 500))
m.lt
m.lt$par

#d)
s.filter=dlmFilter(yt,lt(m.lt$par))
s.filter

# prediction
s.predicted=rep(0,500)
#s.predicted

up <- array(500)
low <- array(500)

for (i in 1:500)
{
  s.predicted[i] =
s.filter$U.R[[i]]%%diag(s.filter$D.R[i,]^2,nrow=1)%%t(s.filter$U.R[[i]])
  up[i]=s.filter$a[i]+2*sqrt(s.predicted[i])
  low[i]=s.filter$a[i]-2*sqrt(s.predicted[i])
}

plot(s.filter$a,type="l",ylim=c(-4,2),xlab="Index",ylab="s",main="predicted
(MLE) with 95% CI",lwd=2)
lines(up,col='blue',lwd=2)
```

```
lines(low,col='red',lwd=2)
lines(yt, col = 'green')

# filtering
s.filtered=rep(0, 500)
s.filtered

filt_up <- array(500)
filt_low <- array(500)

for (i in 2:(501))
{
  s.filtered[i-1] =
s.filter$U.C[[i]]%%diag(s.filter$D.C[i,]^2,nrow=1)%%t(s.filter$U.C[[i]])
  filt_up[i]=s.filter$m[i-1]+2*sqrt(s.filtered[i])
  filt_low[i]=s.filter$m[i-1]-2*sqrt(s.filtered[i])
}

plot(s.filter$m[-1],type="l",ylim=c(-4,2),xlab="Index",ylab="s",main="filtered
 (MLE) with 95% CI",lwd=2)
lines(filt_up,col='blue',lwd=2)
lines(filt_low,col='red',lwd=2)
lines(yt, col = 'green')

# smoothing
s.smooth=dlmSmooth(yt,lt(m.lt$par))

s.smoothed=rep(0,500)

smoothed_up <- array(500)
smoothed_low <- array(500)

for (i in 2:(501))
{
  s.smoothed[i-1] =
s.smooth$U.S[[i]]%%diag(s.smooth$D.S[i,]^2,nrow=1)%%t(s.smooth$U.S[[i]])
  smoothed_up[i]=s.smooth$s[i-1]+2*sqrt(s.smoothed[i])
  smoothed_low[i]=s.smooth$s[i-1]-2*sqrt(s.smoothed[i])
}

plot(s.smooth$s[-1],type="l",ylim=c(-4,2),xlab="Index",ylab="s",main="smoothed
 (MLE)  with 95% CI",lwd=2)
lines(smoothed_up,col='blue',lwd=2)
lines(smoothed_low,col='red',lwd=2)
lines(yt, col = 'green')

#e)

StructTS(yt,type="level",fixed=c(NA,NA))
```