1. To decide what kind of model is appropriate we need to look at the acf and pacf functions. The acf is given, but we have to work the pacf out "by hand." This means applying the Durbin-Levinson algorithm to the acf.

   I tend to be lazy and make mistakes in arithmetic so I decided that it would be easier to write a little R function to do the work. Here it is:

```
durbin.levinson =
function(rho)
{
  n = length(rho)
  pacf = phi = numeric(n)
  pacf[1] = phi[1] = rho[1]
  for(k in 2:n) {
    km1 = k - 1
    numer = rho[k] - sum(phi[1:km1]*rho[km1:1])
    denom = 1 - sum(phi[1:km1]*rho[1:km1])
    phi[k] = numer / denom
    phi[1:km1] = phi[1:km1] - phi[k] * phi[km1:1]
    pacf[k] = phi[k]
  }
  pacf
}
```

   We can apply this function to the pacf as follows.

```
> rho = c(0.41, 0.32, 0.26, 0.21, 0.16)
> durbin.levinson(rho)
[1] 0.41000000 0.18259406 0.09686181 0.04984485
[5] 0.01370679
```

   The standard error for these values is $1/\sqrt{169}$ so that 95% confidence limits are $\pm 1.96/\sqrt{169} = \pm 0.1507692$.

   All the acf values exceed this and appear be decaying exponentially. We can conclude that there is a significant autoregressive component to the model.

The pacf values are more problematic. Only the first two values exceed the confidence limits (sharp cutoff). Together with the acf behaviour this could indicate and AR(2) model. Alternatively, the pacf could be interpreted as exhibiting exponential decay (roughly halving for each extra lag). This could be indicative of an ARMA(1,1) series. I'll accept either argument provided the reasoning is given.

There is an additional check for ARMA(1,1) models, namely that

$$\rho(2)^2 = \rho(1)\rho(3).$$

In this case $r(2)^2 = 0.1024$ and $r(1)r(3) = 0.1066$, which are very close. This means that ARMA(1,1) is probably a suitable model.

2. For the AR(1) model, the acf is given by $\rho(u) = \phi^{|u|}$, and from the notes the variance for $r(u)$ is

$$\text{var } r(u) \approx \frac{1}{T}\left(\frac{(1+\phi^2)(1-\phi^{2u})}{1-\phi^2} - 2u\phi^{2u}\right).$$

To answer the question, I implemented a couple of small R functions as follows

```
rho = function(u, phi = .7) phi^u
sdr = function(u, phi = .7, nobs = 100) {
        sqrt(((1+phi^2)*(1-phi^(2*u))/(1-phi^2)
              -2*u*phi^(2*u))/nobs)
      }
```

and used them to measure the size of differences relative to the standard errors.

```
> (.6 - rho(1))/sdr(1)
[1] -1.40028
> (-.15 - rho(10))/sdr(10)
[1] -1.046114
```

It looks like neither value is particularly surprising.

3. Here is the code I used to answer this question.

```
a1 = numeric(1000)
a2 = numeric(1000)
for(i in 1:1000) {
  x = arima.sim(model = list(ar = .5), n = 100)
  a = acf(x, lag = 2, plot = FALSE)
  a1[i] = a$acf[2]
  a2[i] = a$acf[3]
}
sd(a1)        # var(r(1))
sd(a2)        # var(r(2))
cor(a1, a2)   # var(r(1),r(2))
```

These produce values which are very close to those which come from the asymptotic results given in table 4.1.

Quantity	*Simulation*	*Asymptotic Theory*
$\text{var}(r(1))$	0.0862	$0.087 = 0.87/\sqrt{100}$
$\text{var}(r(2))$	0.116	$0.115 = 1.15/\sqrt{100}$
$\text{cor}(r(1), r(2))$	0.738	0.76