1. (a) We can calculate the type I error as

$$P(x \in \{1,2\}|\theta = 0) = P(x = 1|\theta = 0) + P(x = 2|\theta = 0) = .01$$

And the type II error as

$$P(x \notin \{1,2\}|\theta = 1) = P(x = 3|\theta = 1) = .01$$

(b) We can calculate the p-value for $x = 1$ as

$$\sum_{x:P(X=x|H_0)\leq P(X=1|H_0)} P(X = x|H_0) = P(X = 1|H_0) + P(X = 2|H_0) = .01$$

The Bayes Factor for H_1 vs. H_2 is

$$\frac{P(x = 1|\theta = 1)}{P(x = 1|\theta = 0)} = \frac{.0049}{.005} = .98$$

Qualitatively, this means that $x = 1$ is very, very slightly evidence for H_0 over H_1.

(c) We get a p-value for $x = 2$ of

$$\sum_{x:P(X=x|H_0)\leq P(X=2|H_0)} P(X = x|H_0) = P(X = 1|H_0) + P(X = 2|H_0) = .01$$

But this time the Bayes Factor H_1 vs. H_2 is

$$\frac{P(x = 2|\theta = 1)}{P(x = 2|\theta = 0)} = \frac{.9851}{.005} = 197.02$$

Which is overwhelming evidence for H_1

(d) If we modify our example to

x	1	2	3
$p(x\|\theta = 0)$	0.005	0.005	0.99
$p(x\|\theta = 1)$	0.495	0.495	0.01

Then the p-value for $x = 1$ is still .01, but now the Bayes Factor is

$$\frac{P(x = 1|\theta = 1)}{P(x = 1|\theta = 0)} = \frac{.495}{.005} = 99$$

Which is a very large and in favor of H_1

2. I simulated, for one sided p-values for the coefficient from a simple linear regression, how often a value between .04 and .05 actually corresponded to a true positive. Each p-value was simulated where the true effect β was 0 with probability $1 - \pi$ and drawn from an exponential with coefficient λ with probability π. This was preformed with all the usual assumptions for a linear regression in place. To get each p-value, I simulated as if 10 units received the treatment, 10 received the control, and that underlying variance was 1. This was done by drawing $\hat{\beta} \sim N(\beta, \frac{1}{\sqrt{10}})$, drawing $\hat{s}^2 \sim \frac{\chi^2_{18}}{18}$, and generating the usual p-value from the resulting T score.

For a number of combinations of λ and π, I simulated $100,000$ such p-values, and calculated the portion of such values between .04 and .05 for which the true β was positive. I've include the results below (remember $\mathrm{E}(\beta) = \frac{1}{\lambda}$):

	pi = 0.1	pi = 0.3	pi = 0.5	pi = 0.7	pi = 0.9
lambda = 0.125	0.04	0.12	0.29	0.49	0.81
lambda = 0.5	0.13	0.36	0.57	0.75	0.93
lambda = 2	0.22	0.54	0.74	0.86	0.96
lambda = 8	0.18	0.42	0.62	0.81	0.95
lambda = 16	0.12	0.38	0.58	0.76	0.92
lambda = 32	0.12	0.33	0.56	0.74	0.90

Though I employ a somewhat different simulation, the main result is the Same as Selke et al; a p-value near .05 is not consistently strong evidence for the alternative hypothesis. Indeed, we also see that it provides the best evidence for the alternative with moderate levels of power, since if power is too high one rarely sees p-values near .05 for true positives.

Code:

```
if (problem2) {
    ps  <- c(.1,.3,.5,.7,.9)     # portion positive treatment effect (pi)
    ls  <- c(.125,.5,2,8,16,32) # defines exponential which beta is drawn from
    n  <- 10                     # sample units receive control/treatment each
    boots <- 100000

    # Generates pvalue for each experiment. 1st variable indicates positive effect,
    # 2nd parameter for exponential, 3rd number experimental units
    pval_boot <- function(true,lambda,size) {
        beta      <- ifelse(true,rexp(1,lambda),0)
        beta_hat  <- rnorm(1,beta,n^-.5)
        sigma_hat <- rchisq(1,2*n-2)/(2*n-2)
        tscore    <- beta_hat/sqrt(sigma_hat/n)
        return(pt(tscore,2*n-2,lower.tail=F))
    }

    # For each compination of pi and lambda, generate portion of p-values between
    # .04 and .05 which are true positives over given number boots
    nl<-length(ls)
    np<-length(ps)
    evidence<-matrix(,nrow=nl,ncol=np)
    for (i in 1:nl) {
        for (j in 1:np) {
            l <- ls[i]
            p <- ps[j]
            trues <- runif(boots)<p
            pvals <- unlist(mclapply(trues,pval_boot,lambda=l,size=n,mc.cores=4))
            evidence[i,j] <- mean(trues[.04<=pvals & pvals<=.05])
        }
```

3. To get the posterior, we do the usual Bayesian calculation:

$$\begin{aligned} P(\mu|x) &\propto P(\mu)P(x_1,...,x_n|\mu) \\ &\propto \mu^{m-1}e^{-\lambda\mu}\prod_{i=1}^{n}\frac{\mu^{x_i}}{x_i!}e^{-\mu} \\ &\propto \mu^{m-1+\sum_{i=1}^{n}x_i}e^{-(\lambda+n)\mu}\prod_{i=1}^{n}\frac{1}{x_i!} \\ &\sim Gamma(m+\sum_{i=1}^{n}x_i,\lambda+n) \end{aligned}$$

4. To get the posterior, we do the usual Bayesian calculation, this time substituting in $\tau = \frac{1}{\sigma^2}$:

$$\begin{aligned} P(\tau|x) &\propto P(\tau)P(x|\tau) \\ &\propto e^{-2\tau}\sqrt{\tau}e^{-\frac{1}{2}\tau x^2} \\ &\propto e^{-2\tau}\sqrt{\tau}e^{-\frac{1}{2}\tau x^2} \\ &\propto \tau^{\frac{1}{2}}e^{-(2+\frac{1}{2}x^2)\tau} \\ &\sim Gamma\left(\frac{3}{2},2+\frac{1}{2}x^2\right) \end{aligned}$$

5. Overall, using this method, I had an error rate of 22.3%. The main issue I ran into with the method as described was that a number of allele/population combinations didn't appear in the training set. This isn't necessarily a problem, but with these 0 likelihoods included, some individuals in the test data set had 0 likelihoods of each population. I remedied this by adding one to each of the allele/locus counts in the test data set before calculating frequencies. This corresponds to beginning with a weak uniform prior on the allele/locus frequencies. Also, when an individual was missing allele data at a particular locus, I omitted that locus from likelihood calculation (in the program I just multiplied each likelihood by 1).

Code:

```
subpops <- c("EelR","FeatherHfa","FeatherHsp","KlamathRfa")
# Call given code
source('official/exercises/seeb/train_test.R')

# Function to summarize frequency at locus by population (similar to trainc)
# Add 1 so that no Allele is impossible in each subpopulation
compute_freq <- function(data,locus){
    counts <- table(data[,1+2*locus],data$Population) +
                 table(data[,2+2*locus],data$Population) + 1
    return(counts/sum(counts))
}

# Get frequency at each locus
train_freq <- list()
for (i in 1:12) {
    train_freq[[i]] <- as.data.frame.matrix(compute_freq(train,i))
    train_freq[[i]]$allele <- as.factor(rownames(test_freq[[i]]))
}

# Set uniform prior
priors<-paste(subpops,"prior",sep="_")
test[,priors]<-.25
```

```
# Calculate Log-Likelihood
log_lks <-paste(subpops,"loglk",sep="_")
test[,log_lks]<-0
for (i in 1:12) {
    for (j in 1:2) {
        test <- merge(test,train_freq[[i]],by.x=names(test)[j+i*2],
                        by.y="allele")
        # Missing data won't effect likelihood
        test[is.na(test[,j+i*2]),subpops] <- 1
        test[,log_lks] <- test[,log_lks] + log(test[,subpops])
        test<-test[,!(names(test) %in% subpops)]
    }
}

# Calculate Posterior
posteriors<-paste(subpops,"post",sep="_")
test[,posteriors] <- test[,priors]*exp(test[,log_lks])
test[,posteriors] <- test[,posteriors]/apply(test[,posteriors],1,sum) # normaliz

# Predict population with highest posterior
test$post_max <- apply(test[,posteriors],1,max)
test$predict <- ""
for (pop in subpops) {
    test[test$post_max==test[,paste(pop,"post",sep="_")],"predict"] <- pop
}
```