Laboration 1 - Report

Question 1

The hill-climbing algorithm finds a local optimum. Why we get two different networks as a result from hc is because we get different local optimums for our respective runs depending on what network structure we are starting with.

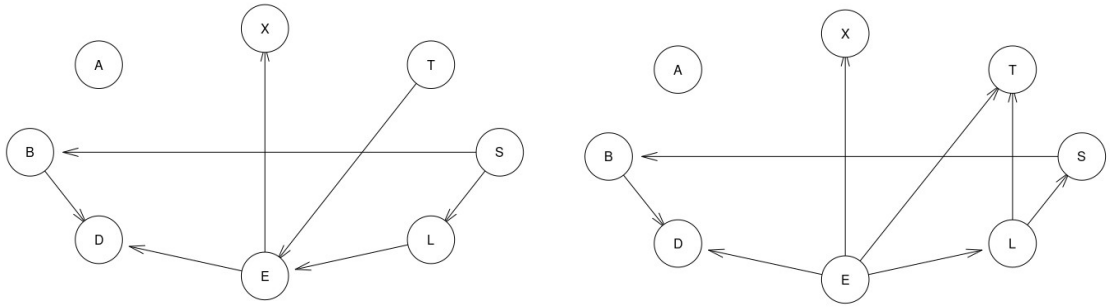
Code:

```
# load library and data
library(bnlearn)
data("asia")

# create the network(s)
bn_1 = model2network("[A][S][T|A][L][B|S][E][X][D|X]")
bn_2 = model2network("[A][L][T|S][S][B|A][D][X|A][E|A]")

# run the hc algorithm on the network
bn_1 <- hc(asia, bn_1)
bn_2 <- hc(asia, bn_2)

plot(bn_1)
plot(bn_2)

arcs(bn_1)
arcs(bn_2)

all.equal(bn_1, bn_2)

cpdag(bn_1)
cpdag(bn_2)
```

Result from all.equal is:

```
> all.equal(bn_1, bn_2)
[1] "Different number of directed/undirected arcs"
>
```

The resulting plots are (bn_1 left, bn_2 right):



Question 2

Method of choice was using exact inference. Running the same code but using the true asia BN gives the same results. The success rate was 0.718.

Code:

```
library(bnlearn)
library(gRain)
data("asia")

set.seed(1)
n <- dim(asia)[1]

# split data
sample <- sample(1:n, floor(n*0.8))
training.data  <- asia[sample, ]
test   <- asia[-sample, ]

s_learn <- hc(training.data)
model <- bn.fit(s_learn, training.data)

junction <- compile(as.grain(model))

predictions <- rep(0, 1000)

for (i in 1:nrow(test)){
  row <- apply(test[i, c(-2)], 2, toString)
  query.result <- querygrain(junction, nodes = "S", evidence = row)
  predictions[i] <- ifelse(query.result$S[1] > 0.5, "no", "yes")
}

c.matrix <- table(test$S, predictions)
success.rate <- sum(diag(c.matrix)/sum(c.matrix))
```

Confusion matrix:

```
> table(test$S, predictions)
     predictions
       no yes
  no  354 174
  yes 130 391
```

Question 3

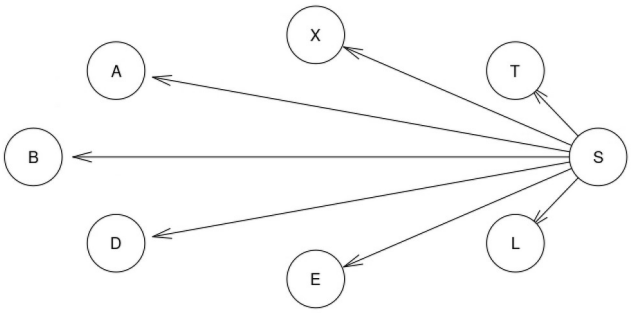
Using mb from bnlearn one gets that the markov blanket consists of [“L”, “B”]. Learning using only these resulted in the same result from question 2 (when using exact inference), both for success rate and the confusion matrix.

```
library(bnlearn)
library(gRain)
data("asia")

set.seed(1)
n <- dim(asia)[1]

# split data
sample <- sample(1:n, floor(n*0.8))
training.data  <- asia[sample, ]
test   <- asia[-sample, ]

s_learn <- hc(training.data)
model <- bn.fit(s_learn, training.data)

mb <- mb(model, "S")

junction <- compile(as.grain(model))

predictions <- rep(0, 1000)

for (i in 1:nrow(test)){
  row <- apply(test[i, c(-1, -2, -3, -6, -7, -8)], 2, toString)
  query.result <- querygrain(junction, nodes = "S", evidence = row)
  predictions[i] <- ifelse(query.result$S[1] > 0.5, "no", "yes")
}

c.matrix <- table(test$S, predictions)
success.rate <- sum(diag(c.matrix)/sum(c.matrix))
```

Question 4

When using the Naive Bayes classifier the results were a little bit worse. Success rate: 0.69.

```
> c.matrix
     predictions
       no yes
  no  381 129
  yes 181 309
```




```r
library(bnlearn)
library(gRain)
data("asia")

set.seed(1)
n <- dim(asia)[1]

# split data
sample <- sample(1:n, floor(n*0.8))
training.data  <- asia[sample, ]
test   <- asia[-sample, ]

naive.bn = model2network("[S][A|S][T|S][L|S][B|S][E|S][X|S][D|S]")
model <- bn.fit(naive.bn, training.data)
plot(naive.bn)

junction <- compile(as.grain(model))

predictions <- rep(0, 1000)

for (i in 1:nrow(test)){
  row <- apply(test[i, c(-2)], 2, toString)
  query.result <- querygrain(junction, nodes = "S", evidence = row)
  predictions[i] <- ifelse(query.result$S[1] > 0.5, "no", "yes")
}

c.matrix <- table(test$S, predictions)
success.rate <- sum(diag(c.matrix)/sum(c.matrix))
```

Question 5

For questions 2 and 3 we get the exact same results. This is because we are basically using the same variables for parameter learning as we're using the variables from the markov blanket and these have the most impact on our variable S.

The difference when we're using the Naive Bayes classifier is because our initial structure is not as good as the ones before. The results are very similar though and this is because our parameter learning is the same as before.