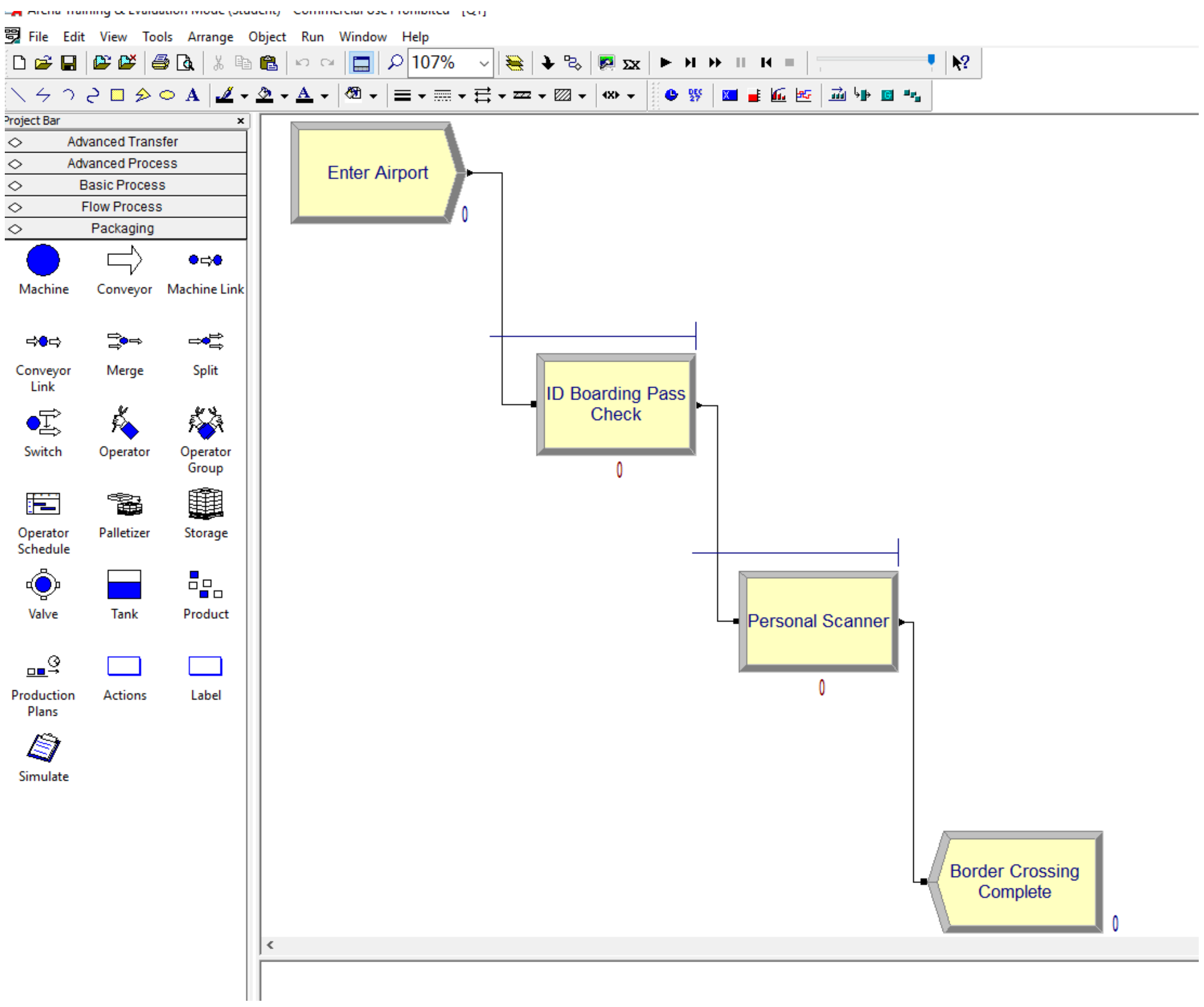
Crab

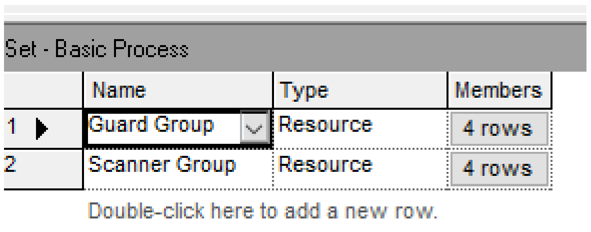
Question 1

I excuted this simulation in Arena over 10 repeats of 5 hours. It was frustrating because I hit a whole lot of errors with the simulation size. With my time over I would complete in SimPy.

Here is my basic setup:



I operated with 4 guards at the Id Check and 4 Scanners:


Set - Basic Process

	Name	Type	Members
1	Guard Group	Resource	4 rows
2	Scanner Group	Resource	4 rows

Double-click here to add a new row.

I pretty much couldnt't reduce resources or the simulation would hit the 150 error.

The results are included in the attached pdf. The end to end average processing time was 0.1184 hours ot a little over 6 minutes.

Question 2

There is a reasonable amount of scripting in this answer, I feel like it could be done much more efficently, in particular where I manual script out the models.

```
# initialize the data
cancer <- read.table('breast-cancer-wisconsin.data.txt', sep=',',header = FALSE)
# head(cancer,10)
# sapply(cancer,class)
# summary(cancer)

cancer.copy <- cancer
cancer.copy$V7[cancer.copy$V7==('?')] <- NA
cancer.copy$V7 <- as.numeric(cancer.copy$V7)
# summary(cancer.copy)

# Pattern of missing values
library('mice')
library(Hmisc)
```

```
## Loading required package: lattice

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked _by_ '.GlobalEnv':
##
##     cancer

## Loading required package: Formula

## Loading required package: ggplot2

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, round.POSIXt, trunc.POSIXt, units
```

```
# how much data is missing
pMiss <- function(x){sum(is.na(x))/length(x)*100}
apply(cancer.copy,2,pMiss) # 2.3% of V7 missing, less than 5%
```

```
##       V1       V2       V3       V4       V5       V6       V7       V8
## 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 2.288984 0.000000
##       V9      V10      V11
## 0.000000 0.000000 0.000000
```

There is 2.3% of the 7th variable missing. We will first sub with the mean and built a model:

```
# impute mean
cancer.mean <- cancer.copy
cancer.mean$V7 <- with(cancer.mean, impute(cancer.mean$V7, mean))
# summary(cancer.mean)

# SVM Classifier
library("kernlab")
```

```
##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
results = {}
#mean dataset
cancer.mean <- as.matrix(cancer.mean)
model.mean <- ksvm(cancer.mean[,1:10],cancer.mean[,11],type="C-svc",kernel="vanilladot",C=100,scaled=TRU
```

```
##  Setting default kernel parameters
```

```
pred <- predict(model.mean,cancer.mean[,1:10])
acc.mean <- sum(pred == cancer.mean[,11]) / nrow(cancer.mean)
results['MEAN'] <- acc.mean
```

Next we built the data set with mode. For this one a defined the function for identifing the mode:

```
# define mode function
getmode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}

# impute mode
cancer.mode <- cancer.copy
cancer.mode$V7 <- with(cancer.mode, impute(cancer.mode$V7, fun=getmode(cancer.mode$V7)))
# summary(cancer.mode)

# mode dataset
cancer.mode <- as.matrix(cancer.mode)
model.mode <- ksvm(cancer.mode[,1:10],cancer.mode[,11],type="C-svc",kernel="vanilladot",C=100,scaled=TRU
```

```
##  Setting default kernel parameters
```

```
pred <- predict(model.mode,cancer.mode[,1:10])
acc.mode <- sum(pred == cancer.mode[,11]) / nrow(cancer.mode)
```

```
results['MODE'] <- acc.mode
```

Here we impute values via regression and then build a model:

```
# imputation via regression
cancer.regr <- cancer.copy
imp <- mice(cancer.regr, method = "norm.predict", m = 1)
```

```
##
##  iter imp variable
##   1   1  V7
##   2   1  V7
##   3   1  V7
##   4   1  V7
##   5   1  V7
```

```
cancer.regr <- complete(imp)

#regression dataset
cancer.regr <- as.matrix(cancer.regr)
model.regr <- ksvm(cancer.regr[,1:10],cancer.regr[,11],type="C-svc",kernel="vanilladot",C=100,scaled=TRU
```

```
##  Setting default kernel parameters
```

```
pred <- predict(model.regr,cancer.regr[,1:10])
acc.regr <- sum(pred == cancer.regr[,11]) / nrow(cancer.regr)
results['REGRESSION'] <- acc.regr
```

I added some random number between -1 and 1 from uniform distribution. There are probably some arguments about the error distribution and scale but I just did this quick and dirty:

```
# add perturbation to regr
cancer.per <- cancer.regr
idx <- which(cancer.copy$V7 %in% NA)
rando <- runif(length(idx),-1,1)
for (i in 1:length(idx)){
  cancer.per[idx[i],7]<-cancer.per[idx[i],7]+rando[i]
}

# use perturbation dataset
cancer.per <- as.matrix(cancer.per)
model.per <- ksvm(cancer.per[,1:10],cancer.per[,11],type="C-svc",kernel="vanilladot",C=100,scaled=TRUE)
```

```
##  Setting default kernel parameters
```

```
pred <- predict(model.per,cancer.per[,1:10])
acc.per <- sum(pred == cancer.per[,11]) / nrow(cancer.per)
results['PERTURBATION'] <- acc.per
```

Here we omit all NA values and just build with the original data:

```
# omit NAs
cancer.orig <- as.matrix(cancer.copy)
model.orig <- ksvm(cancer.orig[,1:10],cancer.orig[,11], na.action = na.omit,type="C-svc",kernel="vanilla
```

```
##  Setting default kernel parameters
```

```
pred.orig <- predict(model.orig,cancer.orig[-idx,1:10])
acc.orig <- sum(pred.orig == cancer.orig[-idx,11]) / nrow(cancer.orig-length(idx))
```

```r
results['REMOVED'] <- acc.orig
```

Here we add a binary value for if there is data missing:

```r
# introduce binary value
cancer.bin <- cancer.copy
cancer.bin$m <- 0
for (i in 1:length(idx)){
  cancer.bin[idx[i],12]<-1
  cancer.bin[idx[i],7]<-0
}
cancer.bin <- as.matrix(cancer.bin)
model.bin <- ksvm(cancer.bin[,-11],cancer.bin[,11], na.action=na.fail, type="C-svc",kernel="vanilladot"
```

```
##  Setting default kernel parameters
```

```r
pred <- predict(model.bin,cancer.bin[,-11])
acc.bin <- sum(pred == cancer.bin[,11]) / nrow(cancer.bin)
results['BINARY'] <- acc.bin
```

And the results are:

```r
results
```

```
##         MEAN         MODE   REGRESSION PERTURBATION      REMOVED
##    0.9699571    0.9685265    0.9685265    0.9685265    0.9470672
##       BINARY
##    0.9699571
```

We see because we fit straight to the data the results are similiar. When we removed the data it become marginally less accurate. We could do smarter things with cross-val and test sets to determine how these imputation methods actually perform but this is ok for now.

Question 3

Optimisation may be used to maximise profits for a car company. They would control the make types according to competitive actions, and vehicle attributes.

We did a piece of work looking to optimise workflows through a major biometric system where the objective function was determined by operator hours required and the number of false matches and false non-matches that proceed through the system.