Construct a data frame x with 1000 columns and 100 rows, and fill it with randomly generated values with the function rnorm().

Question 1

(1/1 point)
You have the existing code below:

```
k<-1000
r<-100
set.seed(5556)
....
```

Which option would replace the .... and perform the task?

- x<-as.data.frame(matrix(rnorm(k),nrow=r))

- x<-as.data.frame(matrix(rnorm(k*r),nrow=k))
- x<-as.data.frame(matrix(rnorm(r*k),nrow=r)) ✔
- x<-as.data.frame(matrix(rnorm(r*k),nrow=k))

Now, construct a summary matrix as follows:

```
my.summary<-matrix(nrow=4,ncol=k)
```

Write a loop that loops over the columns of x, and for each column stores the minimum, median, mean and maximum in the corresponding column of my.summary.

Question 2

(1/1 point)

Drag and drop the fields to the corresponding placement to construct the code

```
for(i in 1:k){
  my.summary[1,i] <- min(x[,i])
  my.summary[2,i] <- median(x[,i])
  my.summary[3,i] <- mean(x[,i])
  my.summary[4,i] <- max(x[,i])
}
```

	[i,1]	[i,2]	[i,3]	[i,4]	

Write a function my.function(), which takes a vector argument, and returns a length-4 vector of minimum, median, mean and maximum of the input. Then recalculate my.summary using sapply().

Question 3

(1/1 point)

You start with the following code:

```
my.function<-function(x){
  ....
}
```

Which option could replace the .... to perform the task?

- return(summary(x))
- return(str(x))
- return(c(max(x),mean(x),median(x),min(x)))
- return(c(min(x),median(x),mean(x),max(x))) ✔

Question 4

(1/1 point)

How would you use sapply() and my.function() to recalculate the result?

- sapply(my.function)

- [x] sapply(x,my.function)
- [ ] sapply(my.function,x)
- [ ] myfunction(x,sapply)

Question 5

(3/3 points)

Calculate the runtime factor using the for loop and compare it to using sapply().

Which operation took more time?

- [x] The for loop
- [ ] The sapply() function

If you increase the data (k and r), let say by 10 folds (either increase k or r by 10 times), and recalculate the runtime factor using the for loop and compare it to using sapply(), which operation took more time now?

- (●) The for loop ✔
- ( ) The sapply() function

If you decrease the data (k and r), let say by 10 folds (either decrease k or r by 10 times), and recalculate the runtime factor using the for loop and compare it to using sapply(), which operation took more time now?

- (●) The for loop ✔
- ( ) The sapply() function