Tutorial 6 sample answer

Question 1

<table>
<tr><td>a)</td><td>Command/code and output:<pre>
> dpois(3, lambda=3)
[1] 0.2240418
</pre></td></tr>
<tr><td>b)</td><td>Command/code and output:<pre>
> ppois(3, lambda=3)
[1] 0.6472319
</pre></td></tr>
<tr><td>c)</td><td>Command/code:<pre>
set.seed(100)
Y <- rpois(100, lambda=4)
mean(Y)
var(Y)
</pre>
Output:<pre>
> mean(Y)
[1] 4.1
> var(Y)
[1] 3.060606
</pre>
Comment:
Using different seed will give different result. Additionally, if we increase the number of sample, mean(Y) and var(Y) will get closer to 4.</td></tr>
<tr><td>d)</td><td>Command/code:<pre>
generate_Z <- function(n){
    X <- rpois(n, lambda=3)
    Y <- rpois(n, lambda=4)
    Z <- X+Y
    return(Z)
}

set.seed(100)
Z <- generate_Z(1000)
mean(Z)
var(Z)

table(Z)/1000   # estimated probability
dpois(7, lambda=7) # theoretical probability
</pre></td></tr>
</table>

	Output:
	``
	> mean(Z)
	[1] 7.148
	> var(Z)
	[1] 7.657754
	> table(Z)/1000
	Z
	    0 1 2 3 4 5 6 7 8
	0.003 0.010 0.016 0.056 0.085 0.105 0.150 0.145 0.139
	    9 10 11 12 13 14 15 16 17
	0.111 0.069 0.049 0.031 0.013 0.005 0.008 0.001 0.002
	   18 19
	0.001 0.001
	> dpois(7, lambda=7)
	[1] 0.1490028
	``
	
	Comment:
	The mean and variance of Z is close to 7, as expected. And if we increase the number of samples, the values will get closer and closer to 7.
	
	From the generated Z, we estimate $P(Z = 7)$ to be 0.145. And as shown, the true value for $P(Z = 7)$ is 0.149.

Question 2

a)	Command/code:
	`f <- function(x){`
	`     return(x^x)`
	`}`

b)	Command/code:
	`monte_carlo_int <- function(n){`
	`     X <- runif(n, min=1, max=2)`
	`     est <- 1/n*sum(f(X))`
	`     return(est)`
	`}`

c)	Command/code:
	`set.seed(100)`
	`monte_carlo_int(100000)`
	
	`integrate(f, 1, 2) # compare using built-in function`
	
	Output:
	`> set.seed(100)`
	`> monte_carlo_int(100000)`
	`[1] 2.048201`
	`> integrate(f, 1, 2)`
	`2.050446 with absolute error < 2.3e-14`
	
	Comment:
	In this question, we attempt to find the estimate of the integral using Monte Carlo method. The integral cannot be solved analytically, and we cannot compare our result with the true/analytical value.
	
	Using Monte Carlo integration, we estimate
	$\int_1^2 x^x \, dx \approx 2.048201$
	using 100 000 samples. As usual, if we increase the number of samples, the estimation will be more accurate, but may become computationally costly. We can compare the result from our Monte Carlo method with the built-in function to integrate numerically in R which gives the value 2.050446, which is not far from our result.

Question 3

	Command/function:

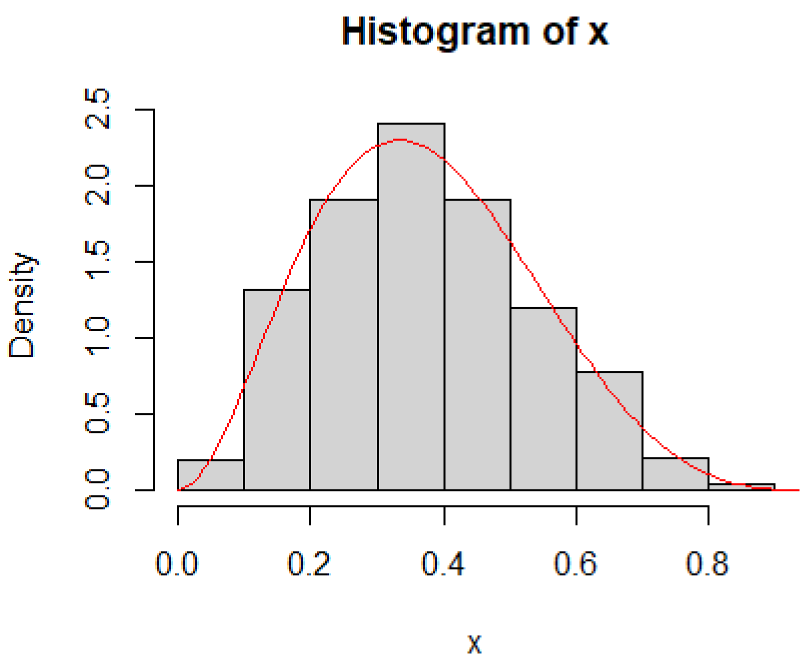
Command/function:

```
generate_x <- function(n){
     x <- c()
     n_accept <- 0
     while(n_accept < n){
          x1 <- runif(1, min=0, max=1)
          u <- runif(1, min=0, max=2.4)
          if(u <= dbeta(x1,3,5)){
               x <- c(x,x1)
          }
     }
     return(x)
}

set.seed(100)
x <- generate_x(1000)
length(x)
hist(x, freq=FALSE)
curve(dbeta(x,3,5), from=0, to=1, col="red",
add=TRUE)
```

Example output:

```
> length(x) #check the size of generated x
[1] 1000
```



	Comment:
	A common mistakes among student is by generating n samples of x_i, and keeping the values of x_i such that $u_i \leq f(x)$. For example:
	
	`Tut6 <- function(n){`
	`    x <- runif(n, 0, 1)`
	`    u <- runif(n, 0, 2.4)`
	`    fx <- dbeta(x, 3, 5)`
	`    samples <- x[u <= fx]`
	`    return(samples)`
	`}`
	
	This is not fully correct because the size of the generated sample will almost always be less than n. In this questions, to have n samples, you will have to use while loop.