Question 1. The following Sage code achieves the desired results. Note that we are using the given formula in class for the probability that no collision occurs. That is,

\[
\Pr[\neg Col(n, k)] = \prod_{i=1}^{k-1} \left(1 - \frac{i}{n}\right).
\]

Note that the code presented is for \(n = 10,000\). Simply repeat with \(n = 100\) for that case. Next, the following plots were created with the data generated by the Sage code. These figures were generated with the \texttt{tikz} package along with the \texttt{pgfplots} package. Note that \(n = 10,000\) has too many data points for \LaTeX{} to process. So it is truncated at \(k = 1,000\), which is fine since the probability is very close to 0.
reset() # resets all variables in the environment

n = 10000  
X = [0]  # stores x-values for plot  
Y = [1]  # stores y-values for plot  

y = (1 - (1/n))  # values for k = 1  
X.append(1)  
Y.append(y)  
k_0 = 0  # for computing k0  
k_1 = n+1  # for computing k1  

# calculates all values for the probability of no collision
for k in range(2,n+1):  
    X.append(k)  
    y = y * (1-(k/n))  
    Y.append(y)  

for k in range(0,n+1):  # calculates k0  
    if Y[k] >= 99/100:  
        if k+1 >= k_0:  
            k_0 = k+1  

for k in range(n, 0, -1):  # calculates k1  
    if Y[k] <= 1/100:  
        if k+1 <= k_1:  
            k_1 = k+1

# prints desired values with k0 and k1 for given n
print "(n,k_0)_={"n","n","k_0","} ; k_0^2/\_n\_={"n","k_0","^2/n\\nprint "(n,k_1)_={"n","k_1","} ; k_1^2/\_n\_={"n","k_1","^2/n\\n
# prepares data for output to a file
st1 = "x_y\r\n"  
for k in range(0,1001):  
    st1 = st1 + str(X[k]) + " " + str(RR(Y[k])) + " \r\n"

# writes data to file for use in LaTeX
o = open('n10000.txt','w')  
o.write(st1)  
o.close()
Question 2. Here is a solution for question 2.

(a) We describe the following security game between an adversary $\mathcal{A}$ and an honest challenger $\mathcal{H}$:
1. Adversary \( \mathcal{A} \) chooses two messages \( m^{(0)} \) and \( m^{(1)} \) of length \( n \) on which they wish to be challenged. \( \mathcal{A} \) sends the messages to \( \mathcal{H} \).

2. Honest challenger \( \mathcal{H} \) picks a uniformly random bit \( b \sim \{0, 1\} \), samples secret key \( sk \sim \text{Gen}(1^\lambda) \), where \( \lambda = n \). \( \mathcal{H} \) picks random \( m \sim \mathcal{M} \), samples \( c \sim \text{Enc}_{sk}(m^{(b)}) \) and \( d \sim \text{Enc}_{sk}(m) \). \( \mathcal{H} \) sends the tuple \( (c, m, d) \) to \( \mathcal{A} \).

3. \( \mathcal{A} \) replies with a bit \( \tilde{b} \in \{0, 1\} \).

4. \( \mathcal{H} \) outputs bit \( z \), where \( z = 1 \) if and only if \( b = \tilde{b} \).

Adversary \( \mathcal{A} \) wins the security game if and only if \( z = 1 \).

\( \text{(b)} \) We show that the one-time pad encryption scheme is completely insecure for this definition. That is, any adversary \( \mathcal{A} \) can predict the bit \( b \) with probability 1. For simplicity, all messages are over \( \{0, 1\}^n \).

Let \( \mathcal{A} \) be any adversary that takes any two distinct \( n \)-bit messages \( m^{(0)} \) and \( m^{(1)} \). \( \mathcal{A} \) sends these messages to \( \mathcal{H} \). \( \mathcal{H} \) samples \( b \sim \{0, 1\} \) and \( sk \sim \text{Gen}(1^\lambda) \), where \( \lambda = n \). \( \mathcal{H} \) picks random \( m \sim \mathcal{M} \). Since this is one-time pad, \( c = m^{(b)} \oplus sk \) and \( d = m \oplus sk \). \( \mathcal{H} \) sends \( \mathcal{A} \) \( (c, m, d) \).

Now, \( \mathcal{A} \) has \( (c, m, d) \). Since this is one-time pad and \( \mathcal{A} \) knows that \( d \sim \text{Enc}_{sk}(m) \), \( \mathcal{A} \) can easily compute

\[
d \oplus m = (m \oplus sk) \oplus m = sk
\]

And thus, \( \mathcal{A} \) gets \( sk \) and can compute

\[
c \oplus sk = (m^{(b)} \oplus sk) \oplus sk = m^{(b)}
\]

So \( \mathcal{A} \) gets \( m^{(b)} \) and simply compares it to their two messages and sends the bit \( \tilde{b} = b \) to \( \mathcal{H} \). Thus, with probability 1, \( \mathcal{A} \) predicts the bit \( b \). So the advantage of \( \mathcal{A} \) is exactly \( 1/2 \). Thus, one-time pad is completely insecure for this definition.