Part 1:

```
library("QuantumOps")
#input wire creation
v <- intket(x=c(0,7),n=3)

#this will be ancilla wires that are both |00>
ancilla <- intket(0,2)

#First CNOT gate between input and ancilla
v <- tensor(v,ancilla)

#These are here to show that if there is an error on any of the qubits
#it can detect the error in the outcome and correct if need be
#g <- tensor(X(),I(),I(),I(),I())
#v <- g%*%v
#print(dirac(v))

#first CNOT gate
g1 <- controlled(gate=X(),n=5,cQubits = 0,tQubit=3)

#Applying CNOT gate
v <- g1%*%v

#start the second CNOT gate
g2 <- controlled(gate=X(),n=5,cQubits=1,tQubit=3)

v <- g2%*%v

#Same thing but with ancilla2

#First CNOT gate between input and ancilla2
g3 <- controlled(gate=X(),n=5,cQubits = 0,tQubit=4)

#Applying CNOT gate
v <- g3%*%v

#start the second CNOT gate
g4 <- controlled(gate=X(),n=5,cQubits=1,tQubit=4)

#apply gate
```

```
v <- g4%*%v

#measure
L <- reduceMeasure(v,3,4,l2r = TRUE)

#output ket
v <- L[[1]]

#output measurement
outcome <- L[[3]]

print(dirac(v))
print("Measurement: ")
print(outcome)
```

With no errors, the output is:

```
> print(dirac(v))
[1] "0.707|000> + 0.707|111>"
> print("Measurement: ")
[1] "Measurement: "
> print(outcome)
[1] 0 0
```

Error on the first qubit:

```
[1] "0.707|011> + 0.707|100>"
> print("Measurement: ")
[1] "Measurement: "
> print(outcome)
[1] 1 0
```

Error on the second qubit:

```
> print(dirac(v))
[1] "0.707|010> + 0.707|101>"
> print("Measurement: ")
[1] "Measurement: "
> print(outcome)
[1] 1 1
>
```

Error on the third qubit:

```
[1] "0.707|001> + 0.707|110>"
> print("Measurement: ")
[1] "Measurement: "
> print(outcome)
[1] 0 1
```

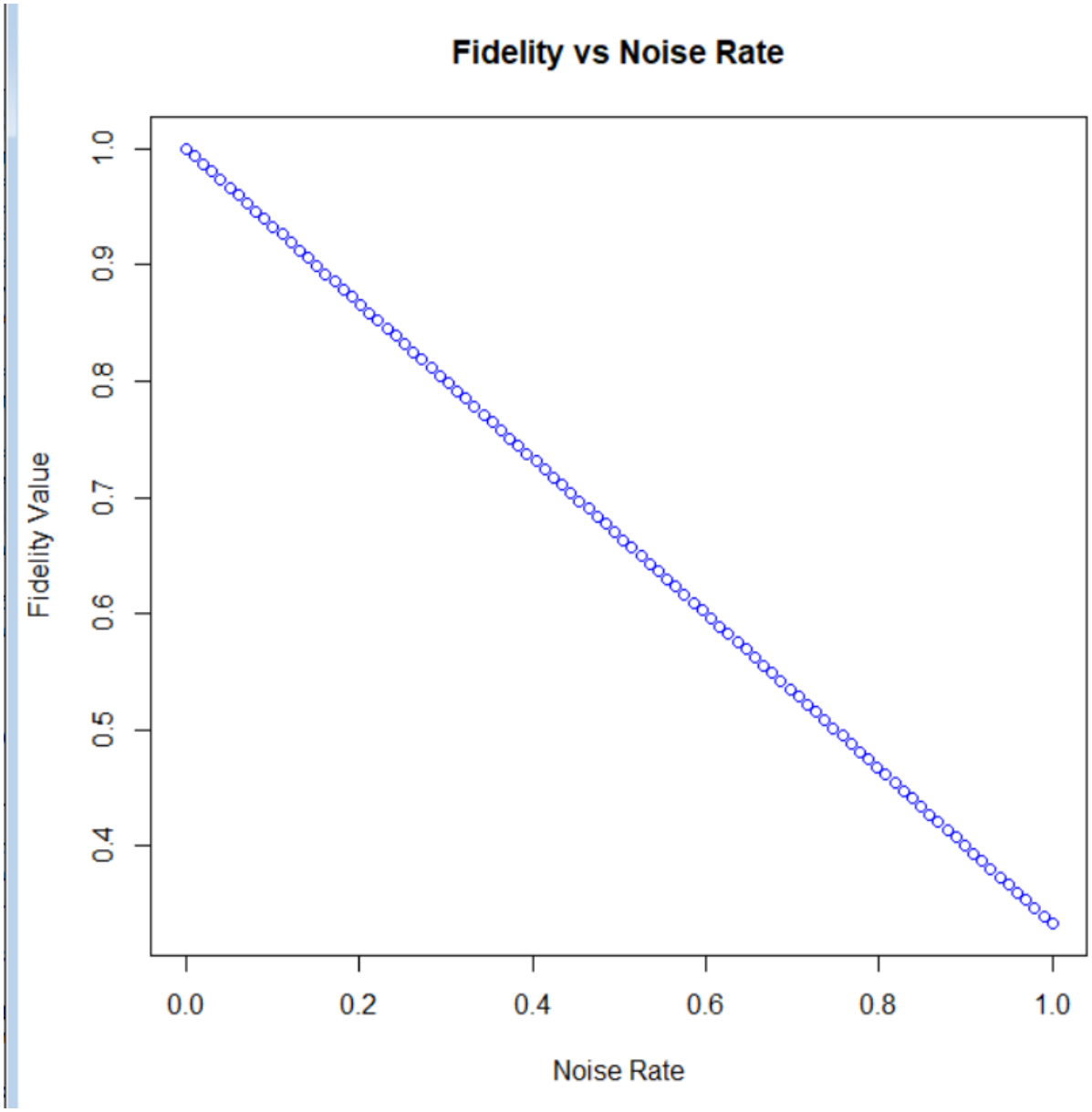
As we can see, the measurement values are different from each other. The method to correct these bits is to look at the measurement output, and change the corresponding bit depending on which output you get. If the output is 0 0, then there is no error. If the output is 1 0, then the error is on the first qubit. If the output is 1 1, the error is on the second qubit. Finally, if you get an output of 0 1, the error is on the third qubit. To fix these errors is to apply an X gate on the respective qubit that is causing the error.

Part 2 Starting Code:

```
library("QuantumOps")
#Function to compute the fidelity of W relative to V (W and V are density
matrices)
fidelity <- function(V,W){
    sum(diag( V %*% W ))
}

#Can convert a ket to density matrix
v <- intket(0,1)

V <- convert_ket2DM(v)

#Density matrices can be subject to more general forms of quantum noise
#Here apply a PauliNoise (of equal amounts X,Y,Z) with a strength of 0.01
W <- PauliNoise(p=V,e=0.01)

print(fidelity(V,W))
```

```
x_axis <- seq(0,1,length.out=100)
y_axis <- numeric(100)
i = 1;
for (val in x_axis){
    W <- PauliNoise(p=V,e=val)
    y_axis[i] <- fidelity(V,W)
    print(fidelity(V,W))
    i = i + 1;
}
plot(x_axis,y_axis,col="Blue",ylab="Fidelity Value",xlab="Noise
Rate",main="Fidelity vs Noise Rate")


#Uncomment me to save
#jpeg("ExamplePlot.jpg")

#Uncomment me to save
#dev.off()
```

Plot:



The fidelity at 1 is 0.33. This drop is due to the imperfect application of quantum gates which can cause noise which will cause a drop in fidelity because of the lack of perfect control over the rotations of the qubits. This value has to do with the tomography of the value of p we found in slide 35. The matrix of p as shown on the slides is p=[0.8 0.4,0.4 0.2] and the value can be found by taking the determinant of it which = 0.32. This value isn't exactly 0.33, but if we were to calculate p using our circuit, we could find the minimum value of fidelity that we want.