Code:

```
library("QuantumOps")
GOracle <- function(w,input){

    #n is set to number of Qubits
    n <- input

    #getting dimensions
    N <- 2^n

    #Build the Oracle matrix
    Uf <- diag(N)

    #setting the diagonals to -1
    Uf[w+1,w+1] <- -1

    #return
    Uf
}

F <- function(x){
    if(x==2){
    return(1)
    }else{
    return(0)
    }
}


f <- function(n,w){
    #Create the ket
    v <- intket(0,n+1)

    #Figure 1: |0> bit Hadamard gate
    H <- H()
    for(j in 2:(n+1))
        H <- tensor(H,H())
    #Applying the H gate to the ket
    v <- U(H,v)
```

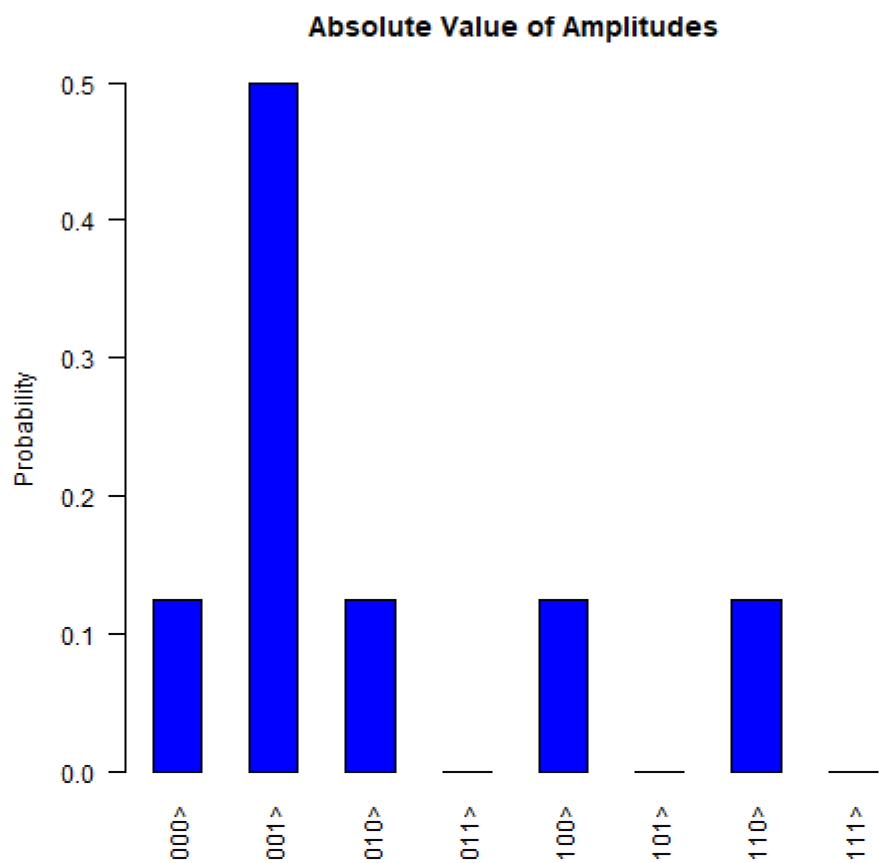
```r
    #Setting up the Grover Oracle
    GO <- GOracle(w,n+1)

    #new GO setup
    #GO <- Uf(F,2,1)

    #Using the built in function to set up the diffusion
    GD <- GroverDiffusion(n)

    #Adding the identity since the last qubit is not affected
    GD <- tensor(GD,I())

    #Testing the barplot function
    P <- rep(NA,6)

    #Loop that applies oracle and diffusion operator 6 times
    for(j in 1:6){
        v <- U(GO,v)
        v <- U(GD,v)

        #Adding the probabilities together for testing
        P[j] <- probs(v)[w+1,1]

        #Outputting those probabilities for checking
        pp("iteration ", j,":", P[j])

        #Plotprobs function I am having problems with
        png( sprintf("ProbabilityDistribution%d.png",j))
        plotprobs(v)
        dev.off()
    }


}

#Calling the function
#first number represents n, second represents the int we are looking for
f(2,1)
```

[redacted]

The output we get from this code is exactly the same result as I get when I manually go through the grover's algorithm, therefore, we know the results are correct. These results make sense if you look at the iteration probabilities in the R terminal:

```
[1] "iteration  1 : 0.5"
[1] "iteration  2 : 0.125"
[1] "iteration  3 : 0.125"
[1] "iteration  4 : 0.5"
[1] "iteration  5 : 0.125"
[1] "iteration  6 : 0.125"
```

We can see here that the highest probabilities happen during iteration 1 and 4. When we look at the png images of these distributions:



We can see here that finding 2 has a very high probability of 50%. This makes sense because grover's algorithm only has a 50% chance to find the correct input, which directly mirrors the values of my outputs.