1. Write a `pois.prob()` function that computes $P(X = x)$, $P(X \neq x)$, $P(X < x)$, $P(X \leq x)$, $P(X > x)$, and $P(X \geq x)$. Enable the user to specify the rate parameter λ.

```r
pois.prob <- function(x, lambda, type){
  # inputs are x, lambda (only parameter of Poison Distribution, and
  #the probability we want to compute
  # lambda = the mean number of events

  # Use dpois and ppois to conditionally return the correct probability

  if (type == "=="){
    # P(X = x) -> pmf
    output = dpois(x, lambda)
  }else if (type == "!="){
    # P(X != x) -> complement rule
    output = 1 - (dpois(x, lambda))
  }else if (type == "<"){
    # P(X < x) -> uses cdf
    output = ppois((x-1), lambda)
  }else if (type == "<="){
    # P(X <= x) -> cdf
    output = ppois(x, lambda)
  }else if (type == ">"){
    # P(X > x) -> cdf + complement rule
    output = 1 - (ppois(x, lambda))
  }else if (type == ">="){
    # P(X >= x) -> cdf + complement rule
    output = 1 - (ppois((x-1), lambda))
  }
  return(output)

}

# Example Run Through
# Question: P( X = x) where x = 0, lambda = 2 following Poison Distribution
pois.prob(x = 0, lambda = 2,"==")


## [1] 0.1353353


# Correct, outputs 0.1353 and that is e^-2 which is the answer
#when calculated analytically
```

The poisson distribution is a discrete distribution that describes the number of events that occur in a unique time/place/space/etc. It takes one parameter as a input: lambda. I used both the PMF and CDF to calculate the operations the question asked for. I validated my answers by using calculus as shown in the comments at the end of my code.

2. Write a `beta.prob()` function that computes $P(X = x)$, $P(X \neq x)$, $P(X < x)$, $P(X \leq x)$, $P(X > x)$, and $P(X \geq x)$ for a beta distribution. Enable the user to specify the shape parameters α and β.

```r
beta.prob <- function(x, alpha, beta, type){
  # alpha = success parameter
  # beta = failure parameter
  # support = [0,1] inclusive
  # Beta Distribution is continuous, not discrete

  # the equals and not equals #
  # Use dbeta and pbeta to conditionally return the correct probability

  if (type == "=="){
    # P(X = x) = 0
    output = 0 # doesn't output the probability
  }else if (type == "!="){
    # P(X != x) = 1
    output = 1 - 0
  }else if (type == "<"){
    # P(X < x) -> uses cdf.      -> same as <=
    output = pbeta(x, alpha, beta)
  }else if (type == "<="){
    # P(X <= x) -> cdf , same as < due to continuous
    output = pbeta(x, alpha, beta)
  }else if (type == ">"){
    # P(X > x) -> cdf + complement rule
    output = 1 - (pbeta(x, alpha, beta))
```

```r
  }else if (type == ">="){
    # P(X >= x) -> cdf + complement rule, same as >
    output = 1 - (pbeta(x, alpha, beta))
  }
  return(output)

}

# Example Run Through
beta.prob(x = 0.4, alpha = 2, beta = 5, ">=")


## [1] 0.23328
```

The beta distribution is a continuous distribution. This allows for easier computation of the basic operations that we were asked to do since P(X ¡= x) is the same as P(X ¡ x) and vice versa. It takes two parameters: alpha and beta which can be seen in the function as inputs. The beta distribution uses the PDF and CDF: `dbeta()` and `pbeta()`. The CDF of continuous functions calculates the cumulative probability.