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.

   Explanation: Here is the code to calculate various probabilities for a Poisson distribution. I changed the `size` and `prob` parameters to λ. The function takes in a support `x` which is greater than or equal 0 and λ which is a value greater than or equal 0. The default `type` for the function is $\leq$.

```r
pois.prob <- function(x, lambda, type="<="){
  # Use dpois and ppois to conditionally return the correct probability

  p.eq <- dpois(x, lambda)     # P(X = x)
  p.neq <- 1 - p.eq            # P(X != x)
  p.lt <- ppois(x - 1, lambda) # P(X < x)
  p.lte <- ppois(x, lambda)    # P(X <= x)
  p.gt  <- 1 - p.lte           # P(X > x)
  p.gte  <- 1 - p.lt           # P(X >= x)

  if (type == "=") {
    return(p.eq)
  }
  else if (type == "!=") {
    return(p.neq)
  }
  else if (type == "<") {
    return(p.lt)
  }
  else if (type == "<=") {
    return(p.lte)
  }
  else if (type == ">") {
    return(p.gt)
  }
  else if (type == ">=") {
    return(p.gte)
  }
}
```

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.

   Explanation: Here is the code to calculate various probabilities for a Beta distribution. I changed the `size` and `prob` parameters to `alpha` and `beta`. The function takes in a support `x` which is $0 \leq$ `x` ≤ 1 and `alpha` and `beta` which are both values greater than or equal 0. The default `type` for the function is $\leq$. The output for $P(X = x)$ and $P(X \neq x)$ were hardcoded as 0 and 1, respectively. The $P(X < x)$ and $P(X \leq x)$ condition and the $P(X > x)$ and $P(X \geq x)$ condition were combined as the equal to becomes irrelevant when considering a continuous distribution.

```r
beta.prob <- function(x, alpha, beta, type="<="){
  # Use dbeta and pbeta to conditionally return the correct probability

  if (type == "="){ #P(X = x)
    return(0)
  }
  else if (type == "!="){ #P(X != x)
    return(1)
  }
  else if(type == "<" | type == "<="){ #P(X < x) or P(X <= x)
    return(pbeta(x, alpha, beta))
  }
  else if (type == ">" | type == ">="){ #P(X > x) or P(X >= x)
    return(1 - pbeta(x, alpha, beta))
  }
}
```