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="<="){
  #check if x is valid for Poisson distribution
  if (x<0 | x != floor(x)){ #check if x is a non-negative integer
    return("Invalid argument for 'x'")
  }
  if (type == "="){ #compute P(X=x) using PMF
    return(dpois(x, lambda))
  }else if(type == "!="){  #compute P(X!=x)
    return(1 - dpois(x, lambda)) # 1 - P(X=x)
  }else if(type == "<"){  #compute P(X<x)
    return(ppois(x -1, lambda)) # P(X<x) = P(X<=(x-1))
  }else if(type == "<="){ #compute P(X<=x) using CDF
    return(ppois(x, lambda))
  }else if(type == ">"){  #compute P(X>x)
    return(1 - ppois(x,lambda)) # P(X>x) = 1 - P(X<=x)
  }else if (type == ">="){  #compute P(X>=x)
    return(1 - ppois(x -1,lambda)) # P(X>=x) = 1 - P(X<=(x-1))
  }else{
    return("Invalid argument for 'type'")
  }
}
```

Poisson distribution is discrete probability distribution. It models the likelihood of a single event occurring within a given time or space. We use `dpois()` function to compute PMF and `ppois()` function to calculate CDF. The parameter `x` cannot be negative and it must be an integer. The function checks if the correct argument for parameters `x` and `type` were passed and the function computes the probability based on the parameter `type`.

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="<="){
  #check if x is valid for Beta distribution
  if (x<0 | x>1){ #check if x is in between 0 and 1
    return("Invalid argument for 'x'")
  }
  if (type == "="){ #compute P(X=x)
    return(0) #continuous distribution, we can measure in finer units
  }else if(type == "!="){  #compute P(X!=x)
    return(1) #continuous distribution, we can measure in finer units
  }else if(type == "<"){  #compute P(X<x)
    return(pbeta(x, alpha, beta)) #P(X<x) = P(X<=x)
  }else if(type == "<="){ #compute P(X<=x) using CDF
    return(pbeta(x, alpha, beta))
  }else if(type == ">"){  #compute P(X>x)
    return(1 - pbeta(x, alpha, beta)) #P(X>x) = 1 - P(X<=x)
  }else if (type == ">="){  #compute P(X>=x)
    return(1 - pbeta(x, alpha, beta)) #P(X>=x) = 1 - P(X<=x)
  }else{
    return("Invalid argument for 'type'")
  }
}
```

Beta distribution is a continuous probability distribution. It models continuous random variables whose range is between 0 and 1. We use `pbeta()` to compute CDF. The function takes a parameter for `x`, which mist be a number between 0 and 1, and the shape parameters `alpha` and `beta`. The function checks if the correct parameters were passed for `x` and `type` and it computes the probability for a desired `type` parameter. Because the distribution is continuous, the function outputs 0 when we request $P(X = x)$ and 1 when we request $P(X \neq x)$.