TASK 13: USE CASE IMPLEMENTATION

AIM:

To develop a Python application that plots four common probability distributions (Binomial, Normal, Poisson, Uniform) by manually calculating their probability formulas and visualizing them with Matplotlib, based on user-provided parameters.

ALGORITHM:

1. **Import Libraries:** Import numpy for numerical arrays, matplotlib.pyplot for plotting, and math for mathematical functions like factorials and exponents.
2. **Define Mathematical Functions:**
   - Create a Python function for each distribution's probability formula (PMF for discrete, PDF for continuous).
   - binomial_pmf(k, n, p): Calculates the probability of k successes in n trials.
   - normal_pdf(x, mu, sigma): Calculates the probability density at point x for a given mean and standard deviation.
   - poisson_pmf(k, lam): Calculates the probability of k events occurring in a fixed interval.
   - uniform_pdf(x, a, b): Calculates the probability density at point x between bounds a and b.
3. **Define Plotting Functions:**
   - Create a separate plotting function for each distribution (plot_binomial, plot_normal, etc.).
   - Each function will generate an appropriate range of x-values and use the corresponding mathematical function from Step 2 to calculate the y-values (the probabilities).
   - Use Matplotlib to create and display a titled and labeled graph of the distribution.
4. **Create Main Function for User Interaction:**
   - Display a menu prompting the user to select a distribution to plot.
   - Based on the user's choice, ask for the required parameters (e.g., mean and standard deviation for Normal).
   - Call the appropriate plotting function with the parameters provided by the user.
5. **Run the Program:** Execute the main function to start the application

PROGRAM:

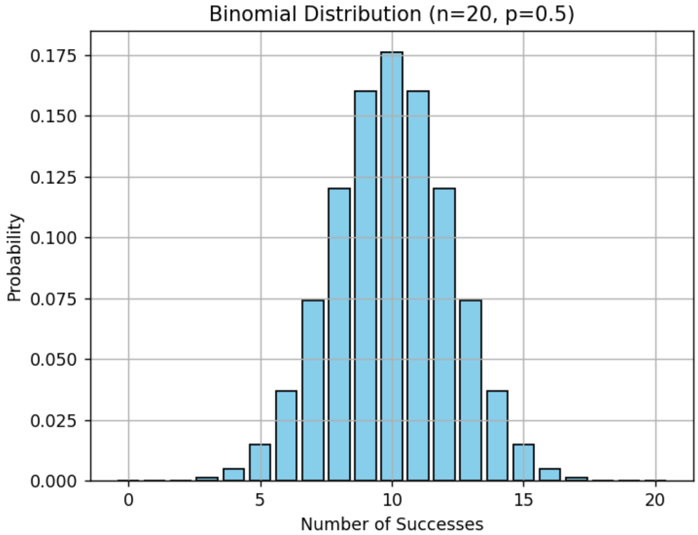
```
import numpy as np
import matplotlib.pyplot as plt
import math

def binomial_pmf(k, n, p):
    if k < 0 or k > n:
        return 0
    combination = math.factorial(n) / (math.factorial(k) * math.factorial(n - k))
    return combination * (p**k) * ((1-p)**(n-k))

def normal_pdf(x, mu, sigma):
```

```
    coefficient = 1 / (sigma * math.sqrt(2 * math.pi))
    exponent = -0.5 * ((x - mu) / sigma)**2
    return coefficient * math.exp(exponent)

def poisson_pmf(k, lam):
    if k < 0:
        return 0
    return (lam**k * math.exp(-lam)) / math.factorial(k)

def uniform_pdf(x, a, b):
    if a <= x <= b:
        return 1 / (b - a)
    else:
        return 0

def plot_binomial(n=10, p=0.5):
    x = np.arange(0, n + 1)
    y = [binomial_pmf(val, n, p) for val in x]
    plt.bar(x, y, color='skyblue', edgecolor='black')
    plt.title(f'Binomial Distribution (n={n}, p={p})')
    plt.xlabel('Number of Successes')
    plt.ylabel('Probability')
    plt.grid(True)
    plt.show()

def plot_normal(mu=0, sigma=1):
    x = np.linspace(mu - 4*sigma, mu + 4*sigma, 1000)
    y = [normal_pdf(val, mu, sigma) for val in x]
    plt.plot(x, y, color='green')
    plt.title(f'Normal Distribution (μ={mu}, σ={sigma})')
    plt.xlabel('x')
    plt.ylabel('Probability Density')
    plt.grid(True)
    plt.show()

def plot_poisson(lam=5):
    x = np.arange(0, 20)
    y = [poisson_pmf(val, lam) for val in x]
    plt.bar(x, y, color='salmon', edgecolor='black')
    plt.title(f'Poisson Distribution (λ={lam})')
    plt.xlabel('Number of Events')
    plt.ylabel('Probability')
    plt.grid(True)
    plt.show()

def plot_uniform(a=0, b=1):
    x = np.linspace(a - 1, b + 1, 1000)
    y = [uniform_pdf(val, a, b) for val in x]
    plt.plot(x, y, color='purple')
    plt.title(f'Uniform Distribution (a={a}, b={b})')
```

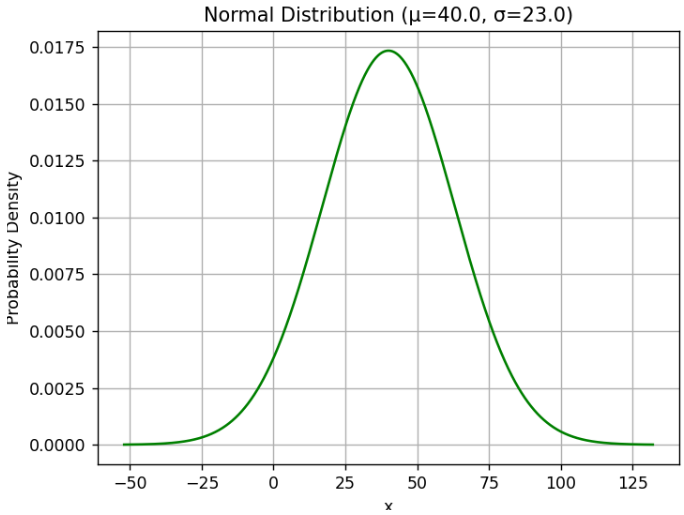
```
    plt.xlabel('x')
    plt.ylabel('Probability Density')
    plt.grid(True)
    plt.show()

def main():
    print("Choose a distribution to plot:")
    print("1. Binomial")
    print("2. Normal")
    print("3. Poisson")
    print("4. Uniform")

    choice = input("Enter choice (1-4): ")

    if choice == '1':
        n = int(input("Enter number of trials (n): "))
        p = float(input("Enter probability of success (p): "))
        plot_binomial(n, p)
    elif choice == '2':
        mu = float(input("Enter mean (μ): "))
        sigma = float(input("Enter standard deviation (σ): "))
        plot_normal(mu, sigma)
    elif choice == '3':
        lam = float(input("Enter rate parameter (λ): "))
        plot_poisson(lam)
    elif choice == '4':
        a = float(input("Enter lower bound (a): "))
        b = float(input("Enter upper bound (b): "))
        plot_uniform(a, b)
    else:
        print("Invalid choice!")

if __name__ == "__main__":
    main()
```

OUTPUT:

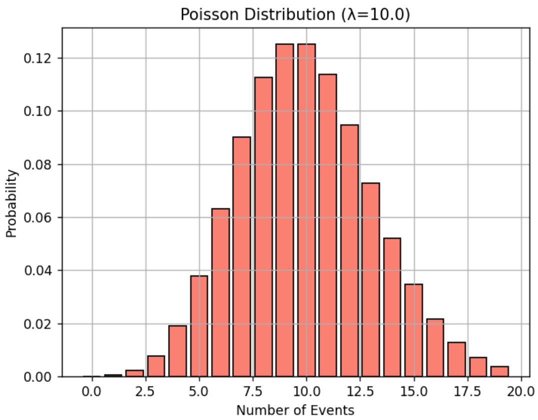
```
Choose a distribution to plot:
1. Binomial
2. Normal
3. Poisson
4. Uniform
Enter choice (1-4): 1
Enter number of trials (n): 20
Enter probability of success (p): 0.5
```



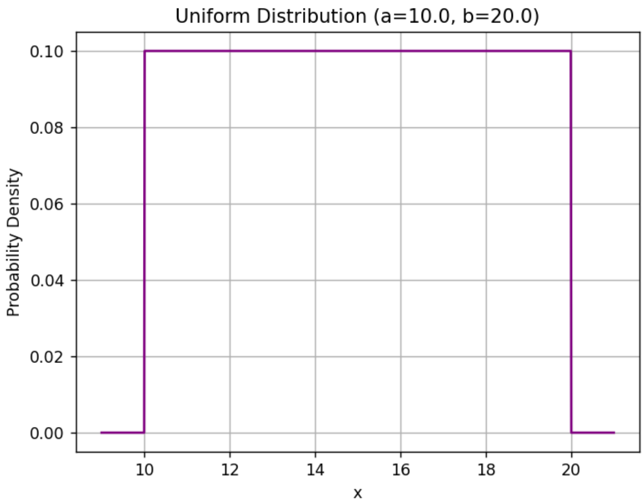

```
Choose a distribution to plot:
1. Binomial
2. Normal
3. Poisson
4. Uniform
Enter choice (1-4): 2
Enter mean (μ): 40
Enter standard deviation (σ): 23
```



```
Choose a distribution to plot:
1. Binomial
2. Normal
3. Poisson
4. Uniform
Enter choice (1-4): 3
Enter rate parameter (λ): 10
```




```
Choose a distribution to plot:
1. Binomial
2. Normal
3. Poisson
4. Uniform
Enter choice (1-4): 4
Enter lower bound (a): 10
Enter upper bound (b): 20
```



RESULT:

The Python application successfully prompts the user to select a probability distribution and provide its parameters. Upon receiving the input, the program correctly calculates the probabilities using fundamental mathematical formulas and generates an accurate, well-labeled plot for the chosen distribution using Matplotlib. The program demonstrates the ability to visualize both discrete (Binomial, Poisson) and continuous (Normal, Uniform) distributions without external statistical libraries.