STA 3180 Statistical Modelling: Markov Chain Monte Carlo

1. Problem: Calculate the probability of a Markov Chain Monte Carlo (MCMC) transition from state A to state B.

```
Solution: Start of Code
// Use the Metropolis-Hastings algorithm to calculate the probability of a transition from state A to state B
// Define the transition probability matrix P
double[][] P = {{0.2, 0.3, 0.5}, {0.4, 0.3, 0.3}, {0.3, 0.4, 0.3}};
// Define the initial state vector x
double[] x = {0.2, 0.3, 0.5};
// Calculate the probability of a transition from state A to state B
double prob = 0;
for (int i = 0; i < 3; i++) {
	prob += x[i] * P[i][1];
}
// Print the result
System.out.println("The probability of a transition from state A to state B is " + prob);
End of Code
```

2. Problem: Implement a Gibbs sampler for a Markov Chain Monte Carlo (MCMC) algorithm.

```
Solution: Start of Code
// Define the transition probability matrix P
double[][] P = {{0.2, 0.3, 0.5}, {0.4, 0.3, 0.3}, {0.3, 0.4, 0.3}};
// Define the initial state vector x
double[] x = {0.2, 0.3, 0.5};
// Initialize the Gibbs sampler
int n = x.length;
double[] x_new = new double[n];
// Iterate over the Gibbs sampler
for (int i = 0; i < n; i++) {
	// Sample from the conditional distribution
	double sum = 0;
	for (int j = 0; j < n; j++) {
		if (j != i) {
			sum += x[j] * P[j][i];
		}
	}
	x_new[i] = sum * P[i][i];
```

```
        // Update the state vector
        x[i] = x_new[i];
}
// Print the result
System.out.println("The updated state vector is " + Arrays.toString(x));
End of Code
```

3. Problem: Implement a Metropolis-Hastings algorithm for a Markov Chain Monte Carlo (MCMC) algorithm.

```
Solution: Start of Code
// Define the transition probability matrix P
double[][] P = {{0.2, 0.3, 0.5}, {0.4, 0.3, 0.3}, {0.3, 0.4, 0.3}};
// Define the initial state vector x
double[] x = {0.2, 0.3, 0.5};
// Initialize the Metropolis-Hastings algorithm
int n = x.length;
double[] x_new = new double[n];
// Iterate over the Metropolis-Hastings algorithm
for (int i = 0; i < n; i++) {
        // Sample from the proposal distribution
        double sum = 0;
        for (int j = 0; j < n; j++) {
                if (j != i) {
                        sum += x[j] * P[j][i];
                }
        }
        x_new[i] = sum * P[i][i];
        // Calculate the acceptance probability
        double alpha = Math.min(1, x_new[i]/x[i]);
        // Sample from the uniform distribution
        double u = Math.random();
        // Update the state vector
        if (u < alpha) {
                x[i] = x_new[i];
        }
}
// Print the result
System.out.println("The updated state vector is " + Arrays.toString(x));
End of Code
```