Question 1:

False

In the multi-armed bandit problem, the value of each arm is not known in advance. The goal is to discover the arm with the highest expected reward through sequential trials. There are various strategies to balance exploration (trying different arms to learn their values) and exploitation (choosing the arm believed to have the highest value to maximize rewards).

The true value of each arm is only revealed after it has been pulled a sufficient number of times to provide an accurate estimate. Until then, the value of each arm is uncertain, and part of the challenge is to manage this uncertainty effectively to maximize cumulative reward over time.

Question 2:

a. which has the highest value when summing an exploitation and and exploration term

The Upper Confidence Bound (UCB) algorithm for the multi-armed bandit problem selects the arm to pull based on a balance between exploitation and exploration. Specifically, it chooses the arm that maximizes the sum of two terms:

1. The exploitation term: This is the estimated value (or average reward) of the arm based on the pulls made so far. The arm with the highest estimated value is the best choice if we were only interested in exploitation.
2. The exploration term: This is a term that increases with the uncertainty or the lack of knowledge about the arm. It also increases when the arm has been pulled fewer times. This term encourages the algorithm to try out arms that have not been pulled as often, to learn more about them.

The UCB algorithm adds these two terms together and chooses the arm with the highest total value, effectively balancing exploitation and exploration.

Question 3:

b. by sampling an arm from a probability distribution that specifies a probability for each arm to be the best one

Thompson Sampling (also known as Bayesian Bandits) is a strategy for the multi-armed bandit problem that works by maintaining a probability distribution for the reward of each arm, based on the results of the pulls made so far. When it's time to choose an arm to pull, Thompson Sampling samples from these distributions, and then chooses the arm corresponding to the highest sample.

This approach naturally balances exploration and exploitation: arms that have not been pulled as often will have wider distributions (reflecting greater uncertainty about their rewards), and so are more likely to produce high samples occasionally. On the other hand, arms that have been pulled frequently and have shown high rewards will have distributions that are tightly concentrated around high values.

Question 4:

True

In a Bernoulli bandit problem, each arm provides a reward that follows a Bernoulli distribution, meaning that the reward is binary: it is either 0 (failure) or 1 (success). The probability of success (getting a reward of 1) is different for each arm, and the goal is to identify the arm with the highest probability of success through sequential trials.

Question 5:

A greedy policy would choose the arm with the highest estimated mean reward, without considering the variance.

From the given options:

- Arm 1: $N(0, 20)$ has a mean of 0
- Arm 2: $N(2, 10)$ has a mean of 2
- Arm 3: $N(1, 1)$ has a mean of 1

Arm 2 has the highest mean reward (2), so a greedy policy would choose:

a. Arm 2: $N(2, 10)$

Question 6:

The Upper Confidence Bound (UCB) action selection rule as described in Sutton & Barto's book in Chapter 2.7 is given by the formula:

$$A_t = \arg\max_a \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

where:

- A_t is the action selected at time t
- $Q_t(a)$ is the estimated value of action a at time t
- $N_t(a)$ is the number of times action a has been selected prior to time t
- c is a constant that determines the level of exploration
- t is the current time step

Let's calculate the UCB values for each arm with the given statistics and $c = 2$ at time step $t = 15$:

1. Arm 1: $Q_t = 4, N_t = 12$
2. Arm 2: $Q_t = 2, N_t = 1$
3. Arm 3: $Q_t = 3, N_t = 2$

We will use these values to calculate the UCB value for each arm and determine which arm the UCB rule would choose.

```
import numpy as np

# Given data
c = 2
t = 15
arms = {
    "a": {"Q_t": 4, "N_t": 12},
    "b": {"Q_t": 2, "N_t": 1},
    "c": {"Q_t": 3, "N_t": 2},
}

# Calculate UCB value for each arm
ucb_values = {arm: data["Q_t"] + c * np.sqrt(np.log(t) / data["N_t"]) for arm, data in arms.items()}
```

ucb_values, max(ucb_values, key=ucb_values.get)
({'a': 4.95009652160578, 'b': 5.291230895031347, 'c': 5.327251684327336}, 'c')

The UCB values for the three arms at time step $t = 15$ with $c = 2$ are calculated as follows:

1. Arm 1 (a): $4 + 2\sqrt{\frac{\ln 15}{12}} \approx 4.95$
2. Arm 2 (b): $2 + 2\sqrt{\frac{\ln 15}{1}} \approx 5.29$
3. Arm 3 (c): $3 + 2\sqrt{\frac{\ln 15}{2}} \approx 5.33$

The UCB rule would choose the arm with the highest UCB value, which is Arm 3 (c) with a UCB value of approximately 5.33.

So the answer to the question is:

c. $Q_t = 3, N_t = 2$

Question 7:

True

Monte Carlo Tree Search (MCTS) is a search algorithm that is widely used in decision-making problems, including game playing and planning. It involves building a search tree and using Monte Carlo simulations to estimate the values of each node in the tree.

The key steps in MCTS are:

1. **Selection**: Starting from the root node, navigate through the tree to select a leaf node using a selection strategy that balances exploration and exploitation.
2. **Expansion**: If the selected leaf node is not a terminal node (i.e., it does not represent a game over or a final decision), create one or more child nodes.
3. **Simulation**: Run a Monte Carlo simulation from the selected node to get an estimate of its value. This involves playing out the game or decision process from the current state to a terminal state using random choices or a default policy.
4. **Backpropagation**: Use the result of the simulation to update the value estimates of the selected node and all its ancestors.

The use of Monte Carlo simulations to estimate values is a defining feature of MCTS, helping the algorithm to deal with the huge state spaces often encountered in decision-making problems without requiring exhaustive search or complete information about the problem.

Question 8:

False

Monte Carlo Tree Search (MCTS) does not strictly dictate the use of a specific policy like UCB or Thompson Sampling at any particular node in the tree. The algorithm is flexible and can be implemented with various selection strategies at different stages of the search process.

The most common selection strategy used in MCTS is UCB (specifically, a variant called UCT, for Upper Confidence bounds applied to Trees), which is often used to balance exploration and exploitation when selecting which node to explore next. However, this doesn't mean that UCB is always used at the root, or that Thompson Sampling is always used at the leaves.

Thompson Sampling could be incorporated into an MCTS algorithm, but this is less common and would depend on the specific implementation and requirements of the task at hand.

So, it would be incorrect to state that MCTS always uses UCB at the root and Thompson Sampling at the leaves. The selection strategies can vary based on the implementation.

Question 9:

In the context of the Monte Carlo Tree Search (MCTS) algorithm as presented in many references, including Sutton & Barto, the four standard operations executed in an MCTS iteration are:

1. **Selection**: Starting from the root node, a path is traversed down the tree by successively selecting the node that appears most promising according to a certain policy (often UCT or another exploration-exploitation balancing strategy) until a leaf node (a node not fully expanded or a terminal state) is reached.
2. **Expansion**: Once a leaf node is reached, and if it's not a terminal state, one or more child nodes are added to expand the tree.
3. **Simulation**: From the newly expanded node, a simulation or rollout is performed by taking actions (often randomly) until a terminal state is reached. This provides an estimate of the value of the node.
4. **Backup**: The result of the simulation is then back-propagated up the tree, updating the value estimates of all nodes along the traversed path.

Therefore, the answer includes:

a. Backup
b. Selection
c. Expansion
d. Simulation

Question 10:

False

Monte Carlo Tree Search (MCTS) generates a search tree that can have branches of varying depths. The algorithm focuses its search on the most promising parts of the tree, which can lead to some branches being explored more deeply than others. This adaptive behavior is one of the strengths of MCTS, as it allows the algorithm to allocate more computational resources to the more critical parts of the search space, potentially leading to better performance in less time.