Problem 1:

Two suggested protocols to increase transaction rate to 70/second are:

1. Increase size of block from 1MB to 10MB.
2. Decrease the block interval time to 1 min.

Issues with increasing block size: This would require a lot more storage, and would make the full nodes more expensive to operate. And eventaully fully operating nodes will decrease, and the system will become more centralized. There might be more double spending attacks due to slower propogation speeds. Achieving consensus would be difficult since validating a block would require lot more efforts.

Issues with decreasing block interval time: Propogation of the block in the network, takes some time. If we reduce the block interval time, block might not be propogated fully in the network till that time. We might not be able to achieve consensus in due time. Block propogation and latency would also lead to more orphan nodes, since they might be delayed in propogating. It would also have an envionmental effect, since power would be used more since block rate is high.

Problem 2:

See folder named q2

Problem 3:

In bitcoin blockchain we can have double spending attacks. Let's see how:

Suppose Alice bought cocaine from Bob by paying him in bitcoin. Seeing this transaction in the most recent block Bob might think that transaction is successful, and would give alway cocaine to Alice. Now if the next random node this is selected in the next round happens to be controlled by Alice. So she might ignore the block including her cocaine transaction, and could build on the second most recent block, not including the cocaine transaction. So next time any miner would see 2 branches of equal length, being honest it would the would be equally probable to build new block on either of them. Since There is no way to distinguish that one of the block tries to double spend, there are approximately 50% chances that double spend would occur.

Problem 4:

(a)

Since the miner is ahead of public blockchain by two secret blocks, all the mining efforts of the rest of the network will be wasted. Other miners would mine on top of what they think is the longest chain, so after the selfish miner announces, that branch would instantly become the new longest chain. Eventaully the rest of block (found by other miners), would become orphan. So in nutshell, gain in selfish mining is that effective share of mining rewards would increase.

(b)

Problem 5:

Problem 6:

Assuming that other miners have detected misbehaving miner's block, the next randomly selected node can "boycott", by not building on top of the misbehaving miner's block. Since there is no real identity in bitcoin blockchain, we cannot really identify the misbehaving node, based on this public key/wallet address. Since detecting misbehaving node is hard, the misbehaving node can simply change his public key, and can again misbehave. He wouldn't be affected much in this case, so "boycott" might not prevent him from misbehaving.

Problem 7:

As a blockchain designer, we can change the puzzle from "find a block whose hash is below certain value" to "find a block for which the hash of a signature on the block is below certain target". So in this case pool manager would have to share his private key with all the pool members, and this would be risky since members might steal money from his wallet. Other alternative would be that pool manager does the signing work and the members compute hash values. But since signing computationally more expensive than computing hash value, this scheme would not work and mining pools would work.

Problem 8:

```
pragma solidity ^0.5.0;

// Our contract name
contract MyFirstContract {
  // denotes account balance as uint
  uint public balance;

  // set balance to be 100 (initial value)
  constructor() public {
    balance = 100;
  }

  // returns the current value of balance paramter of our contract
  function getBalance() public returns(uint) {
    return balance;
  }
}
```

MyFirstContract.sol

```
// method to request a usable contract abstraction for a specific Solidity contract
var MyFirstContract = artifacts.require("./MyFirstContract.sol");

// Include this in exports.
module.exports = function(deployer) {
  // deploy this contract on Ethereum Network
  deployer.deploy(MyFirstContract);
};
```

2_deploy_contracts.js

```
// Export the development configs.
module.exports = {
    networks: {
        development: {
        host: "localhost",      // local ethureum network
        port: 8545,             // Port of operation for ganache
        network_id: "*"         // * to match any network ID, it is a required field
        }
    }
}
```

truffle-config.js

Problem 9:

Pros of Ethereum over Bitcoin are:

1. Ethereum provide us with EVM, which allow code to be verified and executed on the blockchain

But we shouldn't really try to compare Ethereum and Bitcoin on the same scale, as they are targeted for different applications.