**Ans 1.**

A checksum is an error-detection method in a the transmitter computes a numerical value according to the number of set or unset bits in a message and sends it along with each message frame. At the receiver end, the same checksum function (formula) is applied to the message frame to retrieve the numerical value. If the received checksum value matches the sent value, the transmission is considered to be successful and error-free.  
  
A checksum may also be known as a hash sum.

A mismatched checksum shows that the entire message has not been transmitted. TCP/IP and User Datagram Protocol (UDP) provide a checksum count as one of their services.  
  
The procedure of generating checksums from messages is called a checksum function and is performed using a checksum algorithm. Efficient checksum algorithms produce different results with large probabilities if messages are corrupted. Parity bits and check digits are special checksum cases suitable for tiny blocks of data. Certain error-correcting codes based on checksums are even capable of recovering the original data.

**Ans 2.**

* DistributedFileSystem object do a RPC call to namenode to create a new file in filesystem namespace  
  with no blocks associated to it
* Namenode process performs various checks like a) client has required permissions to create a file or  
  not 2) file should not exists earlier. In case of above exceptions it will throw an IOexception to  
  client
* Once the file is registered with the namenode then client will get an object i.e. FSDataOutputStream  
  which in turns embed DFSOutputStream object for the client to start writing data to.       DFSoutputStream handles communication with the datanodes and namenode.
* As client writes data DFSOutputStream split it into packets and write it to its internal queue i.e.  
  data queue and also maintains an acknowledgement queue.
* Data queue is then consumed by a Data Streamer process which is responsible for asking namenode    to allocate new blocks by picking a list of suitable datanodes to store the replicas.
* The list of datanodes forms a pipeline and assuming a replication factor of three, so there will be  
  three nodes in the pipeline.
* The data streamer streams the packets to the first datanode in the pipeline, which then stores the  
  packet and forward it to second datanode in the pipeline. Similarly the second node stores the packet  
  and forward it to next datanode or last datanode in the pipeline
* Once each datanode in the pipeline acknowledge the packet the packet is removed from the  
  acknowledgement queue.
* First, the pipeline is closed, and any packets in the ack queue are added to the front  
  of the data queue so that datanodes that are downstream from the failed node will not  
  miss any packets.
* The current block on the good datanodes is given a new identity, which is communicated to the namenode, so that the partial block on the failed datanode will be deleted if the failed datanode recovers later on.
* The failed datanode is removed from the pipeline, and the remainder of the block’s data is written to the two good datanodes in the pipeline.
* The namenode notices that the block is under-replicated, and it arranges for a further replica to be created on another node. Subsequent blocks are then treated as normal.  
  It’s possible, but unlikely, that multiple datanodes fail while a block is being written. As long as dfs.replication.min replicas (which default to one) are written, the write  
  will succeed, and the block will be asynchronously replicated across the cluster until  
  its target replication factor is reached (dfs.replication, which defaults to three).

**Ans 3.**

* . The pipeline is closed and any packets in the ackqueue are added to the front of the data queue.
* 2. The current block on the good data nodes is given a new identity, which is communicated to the name node
* 3. The failed data node is removed from the pipeline, and a new pipeline is constructed from the two good datanodes.
* 4. The remainder of the block’s data is written to the good data nodes in the pipeline.
* 5. The name node notices that the block is under-replicated, and it arranges for a further replica to be created on another node.
* 6. As long as dfs.namenode.replication.minreplicas (which defaults to 1) are written, the write will succeed.
* 7. The block will be asynchronously replicated across the cluster until its target replication factor is reached (dfs.replication, which defaults to 3).