Exercise 1

a)

The probability is :$1 - P(\text{"no nodes fail"}) = 1 - (1-p)^n$

b)

The probability is $P(\text{"k out of n nodes fail"}) = p^k * p^{n-k}$

c)

We want to show that: $$p_1+p_2+...+p_n=1-(1-p)^n$$ We have preciously determined that the probability that no nodes fails is ($(1-p)^n$) lets call this p_0. If we add this on the lhs and rhs. $$p_0+p_1+p_2+...+p_n=1-(1-p)^n+p_0$$ $$p_0+p_1+p_2+...+p_n=1$$ Which must be true since all probabilities add up to be one in total.

Exercise 3

a)

- What does broadcast provide?
  It provides a way of storring data on the traget machines. To be precise, if a varibale is needed for any kind of comutation on a large rdd it is sent to every host, which will require alot of traffic. Especially if a host manges multiple chuncks of the rdd it will have the date sent multiple times. A broadcast allows for said needed variable to be storred on the host device.

- Which other mechanism does it improve and how?
  It uses a peer to peer communication model to distribute the data among different machines. This speeds up the distribution of tasks because the old way of sending tha data lead to an overall slowdown when using more nodes. Since the computation time fell but the comunication time rose more quickly. The Peer 2 Peer transfer system allowed it to speed up.

- Which features of the distributed program determine the number of times the variable will be actually transmitted over the network?
  It depends on the number of worker nodes. Since each of the nodes will store the variable it will only require the data once.

- Explain the role of tasks and nodes in this:
  The nodes are the receiving machined which will get the data and the task is the reason why we are sending the data in the first place. Both the radcast data and the task are sent to the nodes which then combine both and execute the tasks.

b)

- Describe the function of an accumulator:
  The function of an accumulator is to keep track value of something we want to accumulate during a

task. Therefore an accumulator is created at the master and than put into the function we want to perform in every node. The value of the accumulators is than propagated back to the master.

- What is the alternative implementation without an accumulator and why is an accumulator a preferred option?
  This could also be done using a reduce operation, but it would get somewhat difficult especially when trying to extract multiple of such data.

- Which example application for an acculumator is discussed in the video?
  Its a filter function which sorts out bad records and keeps track of their amount and the amount of bites inside the bad records. Both of these calues are couted using accumulators.

- What has to be implemented in order to define a custom accumulator?
  One has to define a custom AccumulatorParam. Which tells the accumulator how to work with a custom datatype. Therefore it needs a methode to initialize (set a zero) and a methode to merge two custom values in place.

- Compare the accumulator mechanism to the reduce()-function:
  The accumulators can be setup in master and are then executed on the side of a task (- maybe even without a proper task - see forall) while the reduce is a main task on its own and will make it way harder when trying to extract two accumulations like sum and average at the same time.

c)

- Give three examples of RDD operators that result in RDDs with partitioning:
  join() - map() - reduceByKey()

- Explain the connection between partitioning and network trafic:
  If we want to combine data we need to find the corresponding part first. If it is on a different machine it needs to be sent. If wen can however have all our actions be considerate of the same partition we do not need to resend data as we are garantueed that all the partition data will be on the required machine in the process networking requirements are lowered.

- How does the modification on the PageRank example use partitioning to make the code more eficient?
  The links are first partitioned so they only get send to their specific node. Additionally the joins are aware of the partition therefore by step 2 all data need to be sent exactly once saving lots of networking.

- How does Spark exploit the knowledge about the partitioning to save time in task execution?
  Any shuffle will respect the partiotions. This even applies when 2 rdds are combined then at least one of the partitioners is respected.

- How can you create a custom Partitioner?
  One can define a custom subclass of the partitioner, in order to leverage additional knowledge. Three methods need to be implemented. One returning the number of partitions created. One returning the correct partition for a given key. As well as an equals methode, which helps spark understand whether 2 partitioners are compatile or not.