Question 2_2:

a. edf gives better result than bd because if all the values are the smae so if we will use bd we will lose packets with high slack because packets with lower slack will remain in the buffer and will end sooner. For example if we have in the buffer (sized - 4):
(slack - 3, value – 6) (slack - 3, value – 6) (slack - 3, value – 6) (slack - 3, value – 6)
and we recive 2 packets with slack – 6, than they will be dropped (because their value isn't higher)
and after we accept 2 packets from the buffer ,
the remain packets will be dropped (because their slack is – 0)
instead of replacing two packets with higher slack and accept – 4 packets instead of – 2.

b. bd gives better result than edf because if all the slacks are the smae so if we will use edf we will lose packets with high value because packets with lower value will remain in the buffer .
For example if we have in the buffer (sized - 4):
(slack - 3, value – 6) (slack - 3, value – 6) (slack - 3, value – 6) (slack - 3, value – 6)
and we recive 2 packets with value– 12, than they will be dropped (because their slack isn't higher)
and after we accept 2 packets with value – 6 (total 12) from the buffer ,
the remain packets will be dropped (because their slack is – 0)
instead of replacing two packets with higher value
and accept 2 packets with value 12 (total 24 > 12).

c. if we see a lot of packets with higher value than others and low slack we will prefer to use the bd algorithm because we don't want to lose the high values, but if most of the packets have about the same value we will prefer to use edf, also if the packets with higher values have a big slack we will still prefer to use edf.

d. another algorithm is sorting the queue by the time each packet arrived, we give every packet a time: if the packet is in the first line in the input_file so its time is 1 if he in second line so its time is 2 and so on.
If the queue is not full so add packet to the queue otherwise if exists in the queue some packet with lower time than the current packet so drop the packet with lower time and add the current packet to the queue, else if the queue is full drop the current packet.

We can also use fifo, if the queue isn't full accept the packet otherwise drop the packet.

Pros of time algorithm:
its simple and new packets almost always will be accepted, all the old packets are dropped and the new ones get priority, also easy to implement.

Cons of time algorithm:
its not realy good beacuse if the older packets have high value we will loss them and if the newer packets have low slack we will lose them too, therfore this algorithm is not very good.

Pros fifo algorithm: its simple and fair, the older packets are accepting first and all the other dropped, also easy to implement.

Cons of fifo algorithm: its not realy good beacuse if the newer packets have high value we will loss them and if the older packets have low slack we will lose them too, therfore this algorithm is not very good.

Pros of edf:
its always accept the maximum number of packets, wont lose potential packet and overall gives a good result.

Cons of edf:
can lose high value packet with low slack.

Pros of bd:
accepts all packets with high value, can give realy good result.

Cons of bd:
does not consider slack so can get to a situation when result is realy bad (even 0).