Question1:

According to the problem, we have a set of N deliveries and their due date (set T). And we can sort the due dates of these deliveries from earliest to latest such that
T(n) = { T1<T2<T3<T4<T5 ……T(n-1) < Tn }

Base case:

N=1 and since T <= N, we can easily see that there exists two options: 1) T =0, this is an undefined behavior, deliveries will never due on day 0. 2) T = 1, the teleportation machine will deliver the only item on that particular day. Therefore the S is feasible.

Induction Hypothesis:

When N = K, S should be feasible for T such that T<=K, the number of deliveries due within T days in set S should be less or equal to T

Inductive Step:

Need to prove that when N = K+1, S should be feasible for T such that T<=K+1, the number of deliveries due within T days in set S should be less or equal to T. Since we have sorted the T(n) array in an increasing order, and the first K items would be feasibly delivered according to the inductive hypothesis, we would have the (K+1)th element left to deliver. And because the first Kth items are feasibly delivered within T<=K days, so for the (K+1)th element, it would still have at least K+1 - K = 1 days to deliver. Therefore, we would guarantee the (K+1)th element to be delivered within its due date. Therefore, set S is feasible if and only if for all days T <= N.

Question2:

Description of the algorithm:

1) We need to arrange the delivery date array T in ascending order. (can use Merge sort or Heap sort to achieve optimal complexity) → O(nlogn)
2) Sort payment array P according to array T.
3) Find T[k], T[k+1], T[k+2] …… such that they are the deliveries that due on the same day, and sort the corresponding subarray P_{same} in payment array P in a decreasing order that that $P_{same\ a}$ > $P_{same\ b}$ > $P_{same\ c}$ ……
4) At this point we have a array T that is in an ascending order, and an array P with any potential "same day subarray P_{same} " sorted to a descending order.
5) Now we can start iterating through the delivery due date array T starting from index 0.
6) When we encountered a subarray T_{same} such that it contain the same due date X:

a) If the left neighbor of such T_{same} has a due date = x-1, if so, we only include P_{same} [0] of that subarray so we can take the highest payment. Add P_{same} [0] to P_{total}.
b) If the left neighbor of such T_{same} has a due date = x-n, if so, we include P_{same} [0...n]. For example: T = [1,2,5,5,5,5,6,7,8], in this case, we can include the payment for the first 3 items that are due on day 5. $Add\, P_{same}$ [0...n] to P_{total}.
c) Print out P_{total}.

Running Time Analysis:

Using merge/heap sort to sort the original T into the ascending form, and sorting the payment array P into the corresponding order would take O(nlogn). Sorting the the sub-array P_{same} of payment array P would take, in the worst case, O(nlogn), the other operations would generally take some constant time. The overall running time would be O(nlogn)

Proof of correctness:

Base case: N =1, we only have one item to deliver in one day (refer to question 1), and the max payment is fixed → true!

Inductive Hypothesis: N = K, the algo would produce the correct max payment amount.