2.

First, we show that Diverse Subset is NP. Given A, k, and a subset t, we first check if $|t| \geq k$ and we check every pair of customers in t to see if they share the same product. If not, return 'yes', otherwise 'no'. This certifier runs in polynomial time. Then, we show that Set Packing $\leq_P$ Diverse Subset, and since Set Packing is NP-complete, Diverse Subset is NP-complete. We prove like this:

For any instance of Set Packing Problem, X, we have a set U of n elements and a collection $S_1 \dots S_m$of subsets of U, and a number k, we can create an instance of Diverse Subset, Y, like this: We write down every element of U as products and every S_i as customer. If element e_j belongs to S_i, we set A_{ij} to 1, otherwise 0.

If X instance is 'yes', that means we have a collections of at least k of subsets with the property that no two of them intersect. Then we can choose k rows corresponding to these k subsets in problem Y. No two of them will share a same element, thus they are diverse. Y instance is also 'yes'.

If we get 'yes' from a black box that solve Y, we know we can choose at least k customers and no two of the customers share the same product. Thus the k corresponding subsets in X don't share same element. So X instance is also 'yes'. Thus Set Packing $\leq_P$ Diverse Subset.

3.

First, Efficient Recruiting is NP, since for a set of counselors t, we can check if $|t| \leq k$ and if it covers all the n sports with polynomial time. Then we want to prove Set Cover $\leq_P$ Efficient Recruiting. And since Set Cover is NP-complete, Efficient Recruiting is NP-complete. We prove like this:

For any instance of Set Cover X, we have a set U of n elements, a collection $S_1 \dots S_m$ of subsets of U, and a number k. We can convert X to an instance of Efficient Recruiting Y, in the following steps: we treat each of the elements in U as a sport, and S_i as a counselor. If element $e_j \epsilon S_i$ in X, we let counselor i qualify in sport j. Instance Y asks if Efficient Recruiting is possible for at most k counselors.

If X is a 'yes' instance of Set Cover, we will have a collection of at most k subsets and the union of the subsets equals to U. So in Y, we can choose the corresponding k counselors, and they will cover all the n sports, otherwise if i_{th} sport is not covered, it means e_i is not included in the union of the chosen subsets in X.

If we have 'yes' returned from the black box that solve Y, it means we can choose k counselors and their ability cover all the n sports. So the corresponding subsets can cover all the elements in U in problem X. So we prove Set Cover $\leq_P$ Efficient Recruiting.

4.

(a) The general Resource Reservation Problem is NP-complete. It's obviously NP and we want to show Set Packing $\leq_P$Resource Reservation Problem, and prove the latter is NP-complete since Set Packing is NP-complete. We prove like this:

For any instance of Set Packing, X, we have a set U of n elements, a collection of $S_1 \dots S_m$, and a number k. We construct an instance of Resource Reservation Problem, Y out of X. In Y, we have m processes and n resources. If S_i contains element e_j, then the i_{th} process need resource j. Y asks if it's possible to active at least k processes.

If X is a ‘yes’ instance, it means we can select a collection of at least k subsets, that no two of them share the same element. So the corresponding processes in Y will not share resources with each other. It means we can find at least k processes that can be active in the same time.

If Y is a ‘yes’ instance, we can find at least k processes, that share no resources. In X the corresponding subsets will be a collection that no two of them share the same element. So that X is also a ‘yes’ instance. So Set Packing $\le_P$Resource Reservation.

(b) Not NP-complete, algorithm:

```
For I = 1 to n
    For J = 1 to n
        If I≠J and I, J don’t share the same resource
            Return ‘yes’
Return ‘no’
```

To check if I,J share the same resource, we use this routine:

```
Share(I, J):
    For each resource r in I’s resources
        For each resources t in J’s resources
            If r == t
                Return ‘yes’
    Return ‘no’
```

The whole algorithm runs in $O(n^2m^2)$

(c) Not NP-complete. We can solve it with Bipartite Matching algorithm which is polynomial. We have all the people in V1 and all the equipment in V2. For every process, if it requires person i and equipment j, we create an edge (i, j). If the process requires no person, we create a new node in V1 and connect it to the equipment the process requires. If the process requires no equipment, we create a new node in V2 as the same.

We then do a Bipartite Matching, if the max match is at least k, return ‘yes’. Otherwise ‘no’

(d) NP-complete. We can reduce the special case of Set Packing, Independent Set to this problem. For any instance of Independent Set, X, we treat vertexes as processes and edges as resources. If two vertexes share the same resource, they are incident on the same edge. This new instance satisfies that each resource is requested by at most two processes, since an edge can only incident on two vertexes. If X is a ‘yes’ instance, we have a set of vertexes that share no same edge. Then the corresponding processes share no same resources, they can be active at the same time. If Y is a ‘yes’ instance, corresponding vertexes in X is also independent. So Independent Set $\le_P$this special case. It’s NP-complete.

5.

Obviously, Hitting Set is NP, for we can check a set H if its size is at most k and it hit every B_i in polynomial time. We want to prove Vertex Cover $\le_P$ Hitting Set, and since Vertex Cover is NP-complete, we can say Hitting Set is NP-complete. The prove goes like this:

For any instance of Vertex Cover, X, we have a graph G and we can create an instance of Hitting Set, Y. We define the set A in Hitting Set to be vertex set V and a collection $B_1 \dots B_m$ where $m = |E|, B_i = \{u_i, v_i\}\ for\ every\ edeg\{u, v\}\ in\ E$. In Vertex Cover, we want to know if there exist a subset of V contains at more k points that cover all the edge, while In Hitting Set, we want to know if there exist a subset of A, at most k elements, that hits all the B_i.

If X is a ‘yes’ instance, then we choose H to be the corresponding set of vertexes. H must hit every B_i. Conversely, if Y is a ‘yes’ instance, every B_i is hit by H, then the corresponding vertexes in G cover all the edges. So we have Vertex Cover $\leq_P$ Hitting Set.

6.

Monotone Satisfiability with Few True Variables is NP, we can check if an assignment is satisfying and if it uses at most k variables in polynomial time. We want to prove Vertex Cover $\leq_P$ Monotone Satisfiability with Few True Variables.

For any instance of Vertex Cover, X, we have a graph G and we want to know if there exists a subset of at most k vertexes that cover all the edges. We construct an instance of Monotone Satisfiability Y from X like this. For every node, we create a variable and we create clause $(x_i \lor x_j)$ if there’s an edge {i, j} in G. Y wants to know if we have a satisfying assignment that at most k variables are set to 1.

If X is a ‘yes’ instance, it means some nodes $v_{i_1} \ldots v_{i_k}$ cover all the edges in G. We set $x_{i_1} \ldots x_{i_k}$ to 1 and all the clauses should be true since they are all covered by these k variables. Conversely, if Y is a ‘yes’ instance, every clause has at least one variable set to 1. So if we choose the corresponding vertexes of true variables in G, they will cover all the edges. So we say Vertex Cover $\leq_P$ Monotone Satisfiability with Few True Variables. Thus the latter problem is NP-complete.