A)

a) Page 121 of the textbook states that the search version of the Interval scheduling problem can be solved in polynomial time. We also know that if the search version can be solved in polynomial time, so can the decision version. Since it is in polynomial space, it can be reduced to any problem in the NP space i.e, we can solve the problem without any calls to the vertex cover problem. Hence **YES**

b) We have a polynomial time solution for Interval scheduling. We do not have a polynomial solution for Independent Set. Hence if it were possible that we could reduce from Independent Set to Interval scheduling, it would mean that NP = P. Hence **UNKNOWN**

B)

Certifier:

First, we need to show that The Diverse Subset problem is in NP.

```
Certifier_diverse_subset(C,P){
      Let C be set containing all the customers
      Let P be the set containing all the products.
      For every Product pεP
          Count <- Number of customers having product count > 0
          If Count > 1,
              remove all customers using P from C.
          endif
      end for

      if |C| >= k
          return  Yes.
      else
          return No.
}
```

Complexity:

Outer loop : m times.

Counting number of customers is linear hence maximum of n times.

Removing customers - we may have to remove n customers. Hence n times..

Overall complexity = O(m x (n + n)) = O(mn) which is polynomial time

Correctness:

Every iteration, we remove customers that have product p in common. By the end, we have iterated through all products and have hence removed all customers that have any product in common. hence the customers that are left behind, have no products in common. We then count the number of customers and if >=k, return true.

Hence we have a polynomial time certifier and so the problem is NP.

To prove that it is NP-Complete, we try to reduce the **independent set problem** to our diverse subset problem.

Construction:

<u>From independent set to diverse subset:</u>

For every node in the graph, we create a customer and for every edge in G, we create a product. We create an array of Customers Vs Products and if there is an edge into a customer, we say that they customer bought the product corresponding to the edge. Hence, There is an edge between two vertices iff they share a product.
To construct this array, we will have to go through each vertex and edge once. To fill up the array, for each customer(vertex), we will have to scan through all the edges in the worst case. Hence it can be done in mxn time. Overall time is mxn + m+n = O(mn) which is polynomial time.

Reduction:

To prove this, we need to prove that if we find a diverse set of size k in the constructed graph, we can find an independent set of size k in the given graph. If there is a diverse set in the constructed array, it means that there are at least k customers that do not have common products. This means that in the given graph, there are at least k nodes that do not have common edges. But this is the definition for an independent set of size k.

Conversely, we need to prove that if we find an independent set of size k in the constructed graph, there is a diverse subset of size k in the given array. If there is an independent set in the given graph of size k, it means that there are at least k vertices that do not share a common edge. This is only possible if there were at least k customers that do not have a common product in the initial array. But if they do not have a common product, it means that it is a diverse subset of size k. Hence proved.

Hence, we can say that **Independent set ≤p Diverse subset**

As we are able to find a polynomial time reduction from to the independent set problem which is NP complete, we can say that the diverse subset problem is also NP complete.

C) Certifier:

First, we need to show that The Diverse Subset problem is in NP.

```
Let S be the set of all sports and C be the efficient set
of counselors to be verified.
Certifier_efficient_recruitment(){
   If |C| > k
        Return false;
   endif
   For each counselor cεC
     Remove all sports that C is an expert in from S.
   End for.
   If set S is empty
     Return True
   Else
     Return False.
}
```

The loop runs for a maximum of c times and the the inner loop runs for S times overall. Hence overall complexity = O(c) + O(s) which is polynomial
Hence the Efficient recruitment problem is NP.

To prove that the problem is NP Complete, we try to reduce the **Set cover problem** to our efficient recruitment problem.

Construction:

Given a Set Cover, we construct a graph as follows: For each Subset, create a counselor and assign each capability of the subset as a sports expertise for that counselor.
To construct this, we will need to go through all the subsets and all the capabilities for each subset in the worst case. Hence it takes O(mn) where n is number of subsets and m is number of capabilities and is hence polynomial in time.

Reduction:

We need to prove that if we find a set of k counselors that can cover all the sports, we will have a group of sets(not greater than k) that can cover all the capabilities and its converse. In this case, there is no translation of outputs necessary. It is just a change of labeling.

Since we have k counselors that can cover all the groups and each counselor is a subset and the sports are the capabilities, we in fact have a set of k subsets that cover all the capabilities. Conversely, if we have k subsets that cover all capabilities each subset will be equivalent to a counselor and each capability will be equal to a sport and hence we will have k counselors that can cover all the sports. Hence proved.

As **Set Cover ≤p Efficient recruitment** and as the efficient recruitment problem is in NP, it is NP complete.

D)

a) we can write a polynomial time certifier for the problem. When we are given a set of processes, we can count to see if it has a maximum of k elements. We can verify that they do not have any common resource by going through the resource of each of the processes in polynomial time. Hence the problem is in NP.

We can reduce the independent set problem to our general resource reservation problem.

For every vertex in the graph, we create a process and for every edge, we create a resource. A process requires that resource if there is an edge into its corresponding vertex. This can be constructed in polynomial time by scanning once through all the processes and then for each process, scanning all the resources it requires.

Now, to prove the reduction, we need to prove that if there are k processes which do not request the same resources, then there is an independent set of size n and its converse.

Since there are no common resources requests between these processes, there are no common edges between the vertices corresponding to these processes. Hence the vertices form an independent set. Conversely, if there is an independent set of size k, it means that they have no edges in common. This implies that the processes to which these vertices correspond do not have any resource requests in common. Also, if there were a request, then there would have to be an edge.

Hence independent set ≤p general resource reservation

Since the problem is in NP and we can reduce a standard NP complete problem to our problem, our problem is NP complete also.

b) For k = 2, it takes a maximum of O(n^2) steps if we try by brute force. Hence it is polynomial.

```
For every process P_i
     For every process P_j != P_i
          If process do not have a conflict
               return true.
          Endif
     End for
End For
Return false.
```

c) This can be converted into a bipartite graph problem. Create a graph G with a vertex for each person and resource. For every process, add an edge between the person and resource it requires. Now the requirement that a process and resource must be allocated to only one process can be translated to a bipartite matching(with persons in one set and resources in the other). Now, if we get a bipartite matching of size k, return Yes, else no. This is polynomial time.

d) This is a special case of the case (a) with each resource being requested by at most two processes. Hence, the resource is requested only by the vertices it is incident on. It still remains the same and is hence NP complete.

E)

Certifier:

Given a set H, we can count to see if it has a maximum of k elements. For each set S_i , we can see if atleast one element of it is present in H by scanning H and S_i for a total of S_i x H times which is polynomial in time.

Hence the problem in in NP.
We can check for NP completeness by reducing the Set Cover problem to the Hitting set problem.

Construction:

Given a set cover problem, create another set Z such that for every Set S_i, create a point s_i. For every point x_i contained in U, create a set X_i. Create these sets such that if a point xi had been covered by S in the set cover, in Z, the point s_i is now contained in X_i.

To construct this, we will need to scan through each set and for each set the capabilities of the set. Hence it can be completed in polynomial time.

Reduction:

If there is a set of points H, such that every set is represented, the points that are omitted are equivalent to sets that are omitted in the set cover problem. Hence we finally have only the minimal sets that are required to cover all the points. Similarly, if we have a set cover solution, it inversely translates back to the hitting set solution

Hence, we have that **Set Cover problem ≤p Hitting set problem**
As the problem is in NP and a standard NP complete problem can be reduced to it, the hitting set problem is also in NP.