3 Problem 10.3

3.1 Part i)

Create a 2-SAT formula $f(x_1, ...x_k)$ with k variables where x_i represents feature F_i. For any requirement where either F_i or F_k must exist, add a clause $(x_i \vee x_j)$ to f. For any conflicts, add the clause $(\bar{x}_i \vee \bar{x}_j)$. For any dependencies, add the clause $(\bar{x}_i \vee x_j)$
Run $2 - SAT(f)$ and return true iff $2 - SAT(f)$ returns true

Where k is the number of features, It takes $O(|specifications|)$ time to create the boolean formula by adding a clause per specification, and 2-SAT can be solved in time linear to the number of elements. Note that linear by elements $O(k)$ and $O(|specifications|)$ are interchangeable here since there are two elements per clause/specification, $O(|specifications|) \in O(2k) \in O(k)$. Thus, our algorithm runs in linear time, $\in O(k)$.

Since the 2-SAT algorithm has already been proven to be correct, we will prove that there is a bijection between any set of specifications and a 2-SAT formula. Consider all features as boolean variables (either they are included in the final app or they are not) x_i and specifications as sub-functions on pairs of these variables (x_i, x_j) such that a requirement is fulfilled if either x_i or x_j are in the app, a conflict is avoided as long as both x_i and x_j are not on at the same time, and a dependency is satisfied if x_j is included whenever x_i is included. Note that each of these specifications can actually be mapped to boolean 2-variable clauses:

$$requirement(x_i, x_j) \iff (x_i \vee x_j)$$
$$conflict(x_i, x_j) \iff (\bar{x}_i \vee \bar{x}_j)$$
$$dependency(x_i, x_j) \iff (\bar{x}_i \vee x_j)$$

Then, when creating a function with these specifications in mind we get

$$f(F_1, ...F_k) = F(x_i, x_j) \wedge F(x_x, x_y) \wedge ...$$

where $F \in \{(x_i \vee x_j), (\bar{x}_i \vee \bar{x}_j), (\bar{x}_i \vee x_j)\}$ for any $i, j \in [k]$. This is a 2-SAT formula.
Suppose there exists a subset of feautures that satisfy all the specifications. This would mean that an assignment of $x_i, ...x_k \in \{true, false\}$ exists such that all of the clauses in f (which would be specification in PikPok) are true. Thus, if it is possible to satisfy all of the specifications, it must be possible for all of the clauses to be true, so it must be possible to satisfy f, the boolean 2-SAT formula corresponding to the specifications
Now consider if a 2-SAT f corresponding to a certain set of specifications is satisfiable. Then, there must exist an assignment of features $x_i, ...x_k$ such that all the clauses are true, which means that the same assignment of variables must lead to all specifications being true since each clause corresponds to a different

specification and invariably all clauses must be true. Then, a set of specifications is satisfiable if its corresponding 2-SAT formula f is satisfiable.
Therefore, since we have proved both directions, the 2-SAT formula corresponding to a set of specifications is satisfiable iff the set of specifications is satisfiable. Then, since the two are biconditional, the answer to whether the f is satisfiable as returned by the 2-SAT algorithm will answer whether our set of specifications is satisfiable. Thus, our algorithm is correct.

3.2 Part ii)

We claim that a polynomial time algorithm for choosing a subset of features that satisfies the maximum number of feature specifications implies a polynomial time algorithm for max 2-SAT (and therefore SAT). Let $maxPikPok((F_i, F_j, T_1)\ldots)$ be a black box with input as a list of feature specifications with features F_i, F_j and the type of specification T_k. Let $req, conf, dep$ represent requirement, conflict, and dependency types of specifications, respectively.

Now suppose we have a max 2-SAT formula $f(x_1 \ldots x_n)$. Let us maintain a list of specifications, and initialize features $X_1 \ldots X_n$. For each clause in f, it can take the form of both positive literals $(x_i \vee x_j)$, one negated literal $(\bar{x}_i \vee x_j)$, or both negated literals $(\bar{x}_i \vee \bar{x}_j)$. We then append to our list of specifications the specification (X_i, X_j, req), (X_i, X_j, dep), or $(X_i, X_j, conf)$, depending on each of these cases, respectively. The maximum satisfiable 2-SAT clauses is precisely the number of feature specifications returned by $maxPikPok((X_i, X_j, T_k)\ldots)$. The translation is polynomial, only requiring a linear traversal of the 2-SAT formula, so this algorithm for max 2-SAT will be polynomial iff $maxPikPok$ is polynomial. Since SAT is polynomial if max 2-SAT is, this also implies poly-time $maxPikPok$ implies poly-time SAT.

There is a bijection between specifications and 2-SAT clauses. Let x_i be a boolean variable representing if feature X_i is included. The clause $(x_i \vee x_j)$ is true iff either X_i or X_j are in the product, a requirement specification. The clause $(\bar{x}_i \vee x_j) \equiv x_i \Rightarrow x_j$ is true iff X_i being in the product implies X_j also is (a dependency). The clause $(\bar{x}_i \vee \bar{x}_j)$ is true iff both X_i, X_j aren't in the product (a conflict). Hence, a 2-SAT formula can be bidirectionally translated into a valid list of specifications of the same size, with specification i satisfied iff clause i is satisfied.

Indeed, if exactly k specifications are met by a choice of features $\{X_{i_1} \ldots\}$, then take these specifications $S_1 \ldots S_k$ and consider the mapped clauses. These will be satisfied by the above reasoning. So the max 2-SAT value is $\geq k =$ maxPikPok, since at least these k clauses are satisfied. Conversely, if exactly k 2-SAT clauses are satisfied by some assignment of $x_{i_1} \ldots$, then we can map them to the corresponding PikPok specifications, at least which must be satisfied. Thus number of maxPikPok specifications $\geq k =$ max-2 SAT value. Therefore, they are precisely equal. Thus, we have proven the correctness of the reduction.