1) int BellmanFord (Graph *src, Graph *dest){

```
int min = INT_MAX;

int count = 0;
while (g != dest) {
    g->next;
    count += g->distance;
}
if(count < min)
    return count;

}
```

2)
```
bool greedy(S[0..n-1], n) {
    int day = 0;
    for (int i = 0; i < n; i++) {
        input = S[i].d - S[i].p;
        if (input < day)
            return impossible;
        day += S[i].p;
    }
    return possible;
}
```

Greedy has a run-time of $\Theta(n)$.

Worst case input = 1 at every loop.

3) No. The given algorithm is a comparison based algorithm which would mean that it would have a lower bound of $\Omega(n\log n)$, which is not linear. Therefore, we should not believe in Professor X.

4)

```
bool np(V, E, B) {
    int counter = 0;
  for(int i = 0; i < V; i++) {
    for (int j = 0; j < V; j++) {
      if (i != j && i,j ∈ E)        <- if i and j share an edge
        counter++;
    }
  }
        if (counter == B)
            return true;
    return false;
}
```

Cannot be solved in NP, as this has a run-time of $O(V^2)$ and not k^n.

5) V, E, ~~[illegible]~~E

```
bool p(V, E){
    for(int i = 0; i <= V; i++){
        for(int j = 0; j <= V; j++){
            for(int k = 0; k <= V; k++){
                if(i != j != k){
                    if((i,j) e E && (j,k) e E && (i,k) e E)      if i,j,k all share an edge
                        return true;
                }
            }
        }   return false;
}
```

This algorithm has a run-time of $O(V^3)$, which is in polynomial time, and is thus in P.