1. In order to schedule a series of classes to use the fewest lecture halls possible we would first take all the shortest class and put it in a lecture hall and take the next longest and put it in the same hall, as long as they were compatible. If they are not compatible the class would be moved to the next hall. This would continue until the hall was full, then it would continue with the next hall.
   The theoretical run time would be Theta(logn) as it would look at all n classes to make a decision then it would look at n-1 classes etc.

2. To minimize penalties of the jobs we would look at the deadlines, the lowest deadline would go first followed by the next lowest etc. The theoretical run time of this would be Theta (logn) for the same reason as problem one.
3. The process of picking the last activity to start and then picking the next one that is compatible with the previous selections is greedy because it doesn't consider future choices and will not consider options it previously rejected. This yields an optimal solution because it looks at the last activity to start, which would have to be in an optimal solution then looks at the next activity that starts last and is compatible which would also have to be in an optimal solution. This repeats until you arrive at the optimal solution.

4. The program uses two loops in order to find the current max start time that fits with the currently selected classes. In order to do this the first loop passes an array in a function with the second loop.

```
int* answer, int N- number of classes, activity* activities-an array of activities
    for a = 0 to N
            answer[a] = function (activities,N,answer)

function(a,n,an)
    for b=0 to N
            if(an[b] == 0 && a[b].endTime < an[b].endTime)
                    return a[b]
```

The runtime should be Theta(N^2)