Problem 2:

Pre-Implementation Comments:

When reading this question it immediately reminded me of a similar quesiton that was discussed in a lecture in COMP3121

I first wanted to approach this question wtih the following steps:

1. sort by the activity the start time
2. sort by the duration of the activity
3. choose the first item in the sorted list
4. binary search through to the next available start time
5. and continue this until a classroom is completely filled or we're no longer able to fit any more
6. remove the selected activities
7. repeat this k times to fill all k classrooms

However there is an inherent problem with the first two steps, the question that was discussed in COMP3121 sorts these activities by end time, because that essentially is the same thing as sorting by start time and duration, but better because we're not really interested in the start time of an activity unless we want to compare it with the available classrooms.

So after discussing this issue with the tutor the following is now the new implementation strategy:

1. Sort activities by the end time
2. Have a sorted list of the classrooms by the end time (can be implemented with a multiset in C++
3. iteratively go through this sorted list and add it it to the next available classroom where the end time of the classroom is as close as possible to the start time of this current class (this is to minimise the gap between the

activties which essentially maximises the number of activities that a class can have).

4. Keep track of a counter every time this happens and do this until the list of activities is exhausted.
5. Return counter

Pre-Implementation Comments:

This implementation was accepted in the second attempt after I realised I had made a variable name mistake, which essentially showed that my algorithm was correct.