Conference - Hints

Hints to Complete First Part of the Assignment (for upto 70% credit)

1. Either use the heap implementation given in class (heap.c) of a heap of integers or use your own. Test the implementation to make sure it works.

2.Write a simulation function. The simulation function should take in an array of integers representing the times of each talk, in the order the requests were made, the length of that array, and the number of rooms to simulate the conference. The function should return an integer representing how long the conference takes to finish with the given number of rooms. Here is a prototype for the function:

```
int simulate(int* times, int length, int numRooms);
```

For example, if we passed in the array storing [1000, 2000, 1500, 1500, 1300, 2200, 1800, 800], the length of 8 and numRooms = 2, the function should return 6400. If we were to pass in the same information except pass in numRooms = 3, then the function should return 4300. These values represent the sample input worked out in detail in the assignment write up. You should test this function on other test cases that you make up.

This simulate function will create a heap and use it accordingly. It'll add the first numRooms items in the array times to the heap. Then, a second loop will one by one delete the minimum value from the heap and then add a new value corresponding to the time the deleted item comes out plus the amount of time the new lecture takes. Then, you can empty out the heap. The last item deleted is the time the simulation took.

3. Write a binary search function that takes in the same array and length as the function above, but takes in an integer maxTimes, representing the maximum time for running the conference. This function should return the minimum number of rooms such that the conference time does not exceed maxTime. Your binary search function should call the simulate function inside of a while loop, narrowing down the answer to the query. Here is a prototype for this function:

```
int binsearch(int* times, int length, int maxTime);
```

4. Now, just put the pieces together, read in the input, ignoring the names of the lectures, just storing the numbers in the input in order. Then call the binary search function and print its result.

Hints to Complete the Full Assignment

1. Write a heap struct that stores lecture structs in it. A lecture struct should store the following:

Name of the lecture
Duration of the lecture
Start Time of the lecture
End Time of the lecture
Room Number of the lecture

Note that the fourth item listed is redundant (can be calculated from start time and duration), but it's helpful in code to refer to it easily when needed. The extra memory requirement in this case isn't much of a burden.

Thoroughly test your heap struct. I would write it similar to heapexample.c which used an array of pointers. Write a compare function for the struct based on the end time, breaking ties by the room number, as specified in the assignment.

2.Write a simulation function. The simulation function should take in an array of pointers to lectures, representing each lecture, the length of that array, and the number of rooms to simulate the conference. The function should return an integer representing how long the conference takes to finish with the given number of rooms. ***In addition, the function should populate each lecture struct with the appropriate start time, room number and end time for this particular simulation.*** Here is a prototype for the function:

```
int simulate(lecture** talks, int length, int numRooms);
```

Thoroughly test the simulate function before moving on with several examples you make on your own. Intentionally create test cases with the tie-breaker case where multiple rooms (but not all) free up at the same time.

3. Write a binary search function that takes in the same array and length as the function above, but takes in an integer maxTimes, representing the maximum time for running the conference. This function should return the minimum number of rooms such that the conference time does not exceed maxTime. Your binary search function should call the simulate function inside of a while loop, narrowing down the answer to the query. ***After the binary search completes, run the simulate function ONE MORE TIME with the correct number of rooms so that the lecture structs get populated with the correct room numbers, start times and end times.*** Here is a prototype for this function:

```
int binsearch(lecture** times, int length, int maxTime);
```

4. Now, just put the pieces together, read in the input and for the time being, just output the minimum number of rooms. See if this matches. For small cases, output all of the data stored in the room structs and see if the rooms were properly assigned.

5. Once you know the rooms are being assigned properly for small cases, it's time to work on the sort. You can choose Merge Sort, Quick Sort of Heap Sort. Unfortunately, since the compare function for this sort is different, you can't reuse the Heap code easily. (You could with function pointers, but I haven't taught those.) I would recommend one of the other two sorts. The data should already be in an array that you already read in, so you can simply just sort that array. After you sort it, output the data as specified!