Q1. [10+10 pts]

(a) Write a recurrence to describe the running time T(n) of the following function. Solve the recurrence and given a big Oh bound on the running time.

```
MadSummation (A, left, right)
    //Comment: A in an array of integers, of size n [Assume that n is a power of 7]
    size← right-left+1
    IF size ≥ 7
        seventh_part ← FLOOR(size / 7)
        sum←0

        For i←1 to 7
            sum← sum+ MadSummation(A, left, left + seventh_part)
            left ← left + seventh_part
    ELSE
            sum←0
          For i←left to right
                sum← sum + A[i]
    return sum
```

(b) Recall that the Partition (A, p, r) procedure of quick sort returns an index q such that each element of the sub-array A[p… q-1] is less than or equal to A[q] and each element of A[q+1… r] is greater than A[q]. Modify the partition procedure such that it produces two indices q and t, where p<= q<= t<=r, such that

- all elements of A[q… t] are equal to A[q],
- each element of A[p…q-1] is less than A[q], and
- each element of A[t+1…r] is greater than A[q].

First give a brief explanation of your method in English, then give C++ code.

For an n sized array, A, the running time of your method should be O(n).

Also note that p are r are the starting and ending indices of the array.

You cannot call any ready-made function inside your function.

Q2. [5+10+5] The Catalan numbers are defined as follows:

$$C_0 = 1 \quad \text{and} \quad C_{n+1} = \sum_{i=0}^{n} C_i \, C_{n-i} \quad \text{for } n \geq 0;$$

The 0^{th} Catalan number is 1, and the rest are obtained by the recursive formula given above.

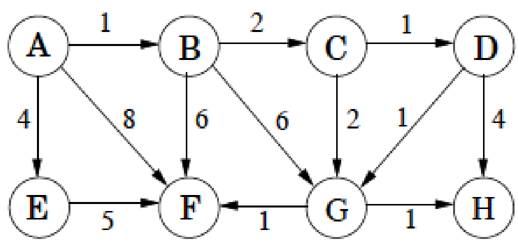
(a) Write a top-down recursive C++ function to compute the nth Catalan number.
(b) Write a bottom-up Dynamic Programming C++ function to compute the nth Catalan number. This function should also allocate the necessary memory required to store sub-problems.
(c) What are the running times in terms of big-Oh of the functions in (a) and (b)

Q3. [6*4] Answer the following questions briefly and top the point, <u>do not write long notes</u>.

(a) Name one feature of quick sort which makes it faster than merge sort in practice. Explain, in at most two lines, why it does so.

(b) If $f(n)=n^{1/2}$ and $g(n)=n^{2/3}$, then which of the following statements is true:
(i) $f(n)=O(g(n))$ (ii) $f(n)=\Omega(g(n))$ (iii) both (i) and (ii)

(c) Suppose Dijkstra's algorithm is run on the following graph. Show the final shortest path tree (take A as the source).



(d) The graph in (c) is also a DAG (directed acyclic graph). Show a linearization of this graph after performing topological sort.

(e) You pay someone k rupees today, and then every day till the end of their life they will keep paying you back half the amount of previous day starting with k/2. (So they will pay you k/2, k/4,…). How many days will it take them to pay back k rupees?

(f) A positive integer is called a perfect number if it is equal to the sum of all of its positive divisors, excluding itself. For example, 6 is the first perfect number, because 6 = 1+2+3. The next is 28 = 14+7+4+2+1. Write a program to find if a number is a perfect numbers. IsPerfect(int num){}

Q4. [5+5+5] Assume you are working in a software house and you are given a new project to work on. Following are the tasks to be done with their duration and dependency.

Activity	Name	Duration (days)	Depends on
A	Account framework	6	
B	Admin login	4	
C	User login	3	A
D	Add course	4	B
E	Remove course	3	B
F	System configuration	10	
G	Logout	3	E,F
H	View course	2	C,D

Starting date of project is 14th Dec. Answer the following questions. Your solution should be optimal.

a) Suppose you want to find the earliest possible project end date. Given what we have studied in class, how will you model this problem? Show it.

b) How will you find the project end date? Explain.

c) Run your algorithm for part (b), show each step in a table.

Note: Rather than re-inventing your algorithm, reuse or transform the algorithms we have studied in class wherever possible.