1. (a) (5%) Please give a precise definition for the big-O notation.

   (b) (5%) When we classify algorithms using the big-O notation, what shall we be aware of, in particular, from the practical aspects of the running time?

2. (a) (3%) Define a stack.

   (b) (12%) Generate a **valid** postfix expression, consisting of 7 symbols, from the symbol set $\{A, B, C, D, *, +, -, /\}$ in a **random** fashion, where A, B, C, and D denote operands, and $*$, $+$, $-$, and $/$ denote operators. A simple two-step algorithm for generating such an expression is as follows:
   (i) Generate a random expression of size 7.
   (ii) Verify the expression generated in Step (i) by using a stack. If the expression is invalid, repeat Step (i); otherwise, terminate the algorithm and return the expression.
   Please implement the two-step algorithm in C or C++, assuming that the functions `rand()`, `pop()`, and `push()` have been defined (yet, the function prototypes for `pop()` and `push()` shall be provided).

3. (a) (3%) Define an AVL tree.

   (b) (3%) Construct the corresponding AVL tree for the ordered integers 4, 13, 19, 22, 31, 53, 57, and 62.

   (c) (4%) Insert the integer 21 into the AVL tree constructed in (b), and form the resulting AVL tree.

4. (a) (3%) Define a max heap.

   (b) (12%) Given an array, `ia`, of n integers in random order, develop an algorithm in C or C++ to rearrange the integers in the array to form a max heap. Assume the function prototype for the algorithm is `void heap(int *ia, int n)`.

5. Answer True or False for the following questions:
   (a) (2%) In a min heap of n nodes, time to delete the smallest node and maintain the heap structure is $O(n \log n)$

   (b) (2%) In a Depth-First-Search tree where each node k is associated with a number order[k] representing the order of DFS labeling. If node i is an ancestor of node j, then order[i]<order[j].

   (c) (2%) In a binary tree, if node X is a descendant of Y, then X always precedes Y in the postorder traversal.

   (d) (2%) When we evaluate a preorder expression + * + 3 4 − 1 2 − 3 / 4 2, we obtain −8.

6. For a connected digraph $G = (N, A)$ of n nodes and m arcs, let $m = o(n^2)$, $n = o(m)$ and c_a denote the length of arc $a \in A$. Answer the following questions about computing single source shortest paths.
   (a) (4%) Suppose G contains directed cycles and each arc has the same length $c > 0$. Can you give a $O(m)$ method to compute shortest paths from a specific source node s to all the other nodes? Briefly explain your answer.

   (b) (4%) Suppose G contains no directed cycles and all arcs have different lengths which are positive. Can you give a $O(m)$ method to compute shortest paths from a specific source node s to all the other nodes? Briefly explain your answer.

   (c) (4%) Suppose G contains directed cycles, all arcs have different lengths, some arcs have negative lengths, but there exist no directed cycles with negative lengths. Thus Dijkstra's algorithm can not be directly applied. Suppose $C = \min_{a \in A}\{c_a\} < 0$. To solve a single source shortest path problem in G from a source node s, someone suggests the following method:

   Algorithm Modified_Dijkstra
   1.**begin**
   2. **for** $i = 1$ to m **do**
   3. $c_i = c_i + |C|$;
   4. do Dijkstra's algorithm starting from source node s with the updated arc length c
   5.**end**

   Will this method solves the original problem? If yes, explain why; otherwise, explain and give a counter example.

7. Rank the following functions in decreasing order of their growth rates
   (a) (5%) $1000^{\log\log n}$, $n!$, n^3, 1000^{n-1000}, $(1.001)^{(\log n)^3}$

   (b) (5%) $0.0000001(\log n)^2$, $0.001n^{0.0001}$, $1000000\log\log n$, $1000(\log n)(\log\log n)$

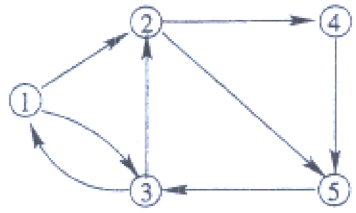

Figure 1: a sample directed graph

point
1
3
5
7
8
9

arc	tail	head
1	1	2
2	1	3
3	2	4
4	2	5
5	3	1
6	3	2
7	4	5
8	5	3

trace
5
1
6
2
8
3
4
7

rpoint
1
2
4
6
7
9

Table 1: CFRS representation of the digraph in Figure 1

8. To represent a connected digraph $G = (N, A)$ of n nodes and m arcs, we introduce "Compact Forward and Reverse Star Representation" (CFRS) which uses 2 arrays (**point**[·], **rpoint**[·]) of size $n+1$, and 3 arrays (**tail**[·], **head**[·], **trace**[·]) of size m where arcs are numbered from 1 to m by nondecreasing order of their tails. In particular, arc[i] points from tail[i] to head[i]; point[k] indicates the smallest numbered arc in the arc list that goes out from node k. Thus we store the outgoing arcs of node k at positions point[k] to point[$k+1$]-1. Likewise, if we sort the arcs in the nondecreasing order of their heads, we obtain rpoint[k] which stores the smallest numbered arc in the arc list that comes into node k. Since point[·] and rpoint[·] use different arc ordering, we use array trace[·] to map one ordering to the other.

   For example, Table 1 lists the CFRS data structures for the graph in Figure 1. It shows that node 3 has 2 (=point[4] - point[3]) outgoing arcs with index 5 and 6; node 2 has 2 (=rpoint[3] - rpoint[2]) incoming arcs with original index trace[rpoint[2]]=1 and trace[rpoint[2] + 1]=6; node 5 has 2 (=rpoint[6] - rpoint[5]) incoming arcs with original index trace[rpoint[5]]=4 and trace[rpoint[5] + 1]=7.

   For a digraph $G = (N, A)$ of n nodes and m arcs, its $n \times n$ adjacency matrix **A**[·,·] is defined by A[i, j]=1 if arc $(i, j) \in A$, and 0, otherwise. Answer the following questions:

   (a) (5%) Suppose you are given the CFRS data structures (i.e. point[·], rpoint[·], tail[·], head[·], and trace[·]) for G. Write an $O(m)$ time pseudocode which converts the CFRS data structures into the $n \times n$ adjacency matrix A[·,·].

   (b) (15%) Suppose you are given the $n \times n$ adjacency matrix A[·,·] for G. Write a pseudocode which converts A[·,·] into CFRS data structures point[·], rpoint[·], tail[·], head[·], and trace[·] in $O(n^2)$ time.