所別：資訊工程學系碩士班　科目：資料結構與演算法
軟體工程研究所碩士班

1. We present an outline of an algorithm to convert an infix string without parentheses into a postfix string below. Where "stk" is a stack; "stacktop(stk)" returns the top element of a stack without removing it from the stack stk; "pop(stk)" pops an element from the stack stk; "push(stk, symb)" pushes the element symb onto the stack stk; "empty(stk)" returns TRUE if the stack stk is empty and FALSE if it is not empty. At the same time, let's assume the existence of a function precede(op1, op2), where op1 and op2 are characters representing operators. This function returns TRUE if op1 has precedence over op2 when op1 appears to the left of op2 in an infix expression without parentheses. precede(op1, op2) returns FALSE otherwise. For example, precede('*', '+') and precede('+', '+') are TURE, whereas precede('+', '*') is FALSE.

```
1     stk = the empty stack;
2     while (not end of input) {
3           symb = next input character;
4           if (symb is an operand)
                add symb to the postfix string;
5           else {
6               while(!empty(stk) && precede(stacktop(stk), symb)) {
7                   push(stk, symb); }
8               topsymb = pop(stk);
9               add topsymb to the postfix string; }}
10    while (!empty(stk)) {
11              push(stk, symb); }
```

(a) (15%) There are errors in this algorithm. Please correct them. You must first identify the line number and then correct them, otherwise you get zero score.

(b) (10%) What modification must be made to this algorithm to accommodate parentheses? Note that the answer is, surprisingly, little. That is, a certain line is replaced by the statement:

```
x     if (empty(stk) || symb != ')')
          push(stk, symb);
      else /* pop the open parenthesis and discard it */
          topsymb = pop(stk);
```

where "x" above is a line number. Of course, the precedence rules for the parentheses should be correctly set. Please set TRUE or FALSE for the following two rules:

precede('(', op)　for any operator op

precede(op, ')')　for any operator op other than '('

2. (10%) Find the shortest path from A to H in the following directed network (its edges with weights are shown below).

AB 14; AC 5; AE 7; BD 4; CB 13; CF 16; DF 3; DH 18; EB 6; FH 11

注：背面有試題

3. (10%) Use stack to do depth-first traversal over the following undirected graph (its edges are shown below) starting at A.

   AB; AC; AE; BD; CB; CF; DF; DH; EB; FH

   Show the vertices visited during the traversal. Note that the neighbors of a vertex should be processed in alphabetical order

4. (5%)

   (a) Draw a sketch to show the doubly-linked implementation of an empty list.

   (b) Insert the element "John" to the list. Show the sketches.

5. (15%) An undirected graph $G=(V, E)$ is stored in a text file with the following format: The first line contains two integer numbers n and m that denote the numbers of vertices and edges of G respectively. Then, the first line is followed by m lines. Each line contains two distinct integers, say i and j, indicating that there is an edge between vertices i and j. Given such a file, design an $O(n)$ time algorithm to test if the undirected graph represened by the file is a tree. You should specify the data sture used to store the graph in the memory and how you construct such a data structure.

6. (15%) Consider a city whose streets are defined by an $m \times n$ grid. We are interested in walking from the upper left-hand corner of the grid to the lower right-hand corner. However, the city has bad areas, which are defined as intersections of streets we don not want to walk in. We are given an $m \times n$ matrix BAD, where BAD[i,j] = "yes" if the intersection between streets i and j is somewhere we want to avoid. Give an $O(m \cdot n)$ time algorithm to find the shortest path across the grid that avoids bad areas. You may assume that all blocks are of equal length.

7. Suppose that k workers are given the task of scanning through a shelf of books in search of a given piece of information. To get the job done efficiently, the books are to be partitioned among k workers. To avoid the need to rearrange the books, it would be simplest to divide the shelf into k regions and assign each region to one worker. Each book can only be scanned by one worker. You are asked to find the fairest way to divide the shelf up. For example, if a shelf has 9 books of sizes 100, 200, 300, 400, 500, 600, 700, 800 and 900 pages, and $k = 3$, the fairest possible partition for the shelf would be:

   100 200 300 400 500 | 600 700 | 800 900

   where the largest job is 1700 pages and the smallest job 1300. In general, we have the following problem: Given an arrangement S of n nonnegative numbers, and an integer k, partition S into k regions so as to minimize the difference between the largest and the smallest sum over all regions.

   (a) (5%) Give an $O(n)$ time algorithm to solve the problem for the case $k = 2$.

   (b) (10%) Design an efficient algorithm to solve the problem for the case $k = 3$. Analysis the time efficiency of your algorithm. (The score you get depends on the efficiency of the algorithm.)

   (c) (5%) Extend your algorithm to the more general case for any given $k \leq n$.