Suggested Solutions to Midterm Problems

1. Prove by induction that any tree can be colored with two colors such that each parent is in a different color from its children.
   Solution. The proof is by induction on the number n of nodes in the tree.

   Base case: When $n = 1$, the only node can be colored in either of the two colors.

   Induction step: Consider a tree T with $n = k + 1$ ($k \geq 1$) nodes. We select a leaf node v in T and delete v (along with the edge connecting v to its parent) from T to obtain a tree T' with k nodes. By the induction hypothesis, T' can be colored with two colors such that each parent is in a different color from its children. Now, color node v in a different color from its parent. Since v is the parent of no other nodes, we have successfully obtained a proper coloring for T. This completes the induction step. $\square$

2. Construct a gray code of length $\lceil \log_2 18 \rceil$ $(= 5)$ for 18 objects. Show how the gray code is constructed from gray codes of smaller lengths.
   Solution. Let $(c_1, c_2, \ldots, c_n)^R$ denote the list $c_n, c_{n-1}, \ldots, c_1$.

   Code of length 1 for 2 objects: $0, 1$.
   Code #1 of length 2 for 2 objects: $00, 01$.
   Code #2 of length 2 for 2 objects: $10, 11$.
   Code of length 2 for 4 objects: $00, 01, (10, 11)^R$.
   Code of length 2 for 4 objects: $00, 01, 11, 10$.
   Code #1 of length 3 for 4 objects: $000, 001, 011, 010$.
   Code #2 of length 3 for 4 objects: $100, 101, 111, 110$.
   Code of length 3 for 8 objects: $000, 001, 011, 010, (100, 101, 111, 110)^R$.
   Code of length 3 for 8 objects: $000, 001, 011, 010, 110, 111, 101, 100$.
   Code of length 4 for 9 objects: $0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100,$ **1100**.
   (open)
   Code #1 of length 5 for 9 objects: $00000, 00001, 00011, 00010, 00110, 00111, 00101, 00100,$ **01100**.
   Code #2 of length 5 for 9 objects: $10000, 10001, 10011, 10010, 10110, 10111, 10101, 10100,$ **11100**.
   Code of length 5 for 18 objects: $00000, 00001, 00011, 00010, 00110, 00111, 00101, 00100, 01100,$ $(10000, 10001, 10011, 10010, 10110, 10111, 10101, 10100, 11100)^R$.
   Code of length 5 for 18 objects: $00000, 00001, 00011, 00010, 00110, 00111, 00101, 00100, 01100,$ $11100, 10100, 10101, 10111, 10110, 10010, 10011, 10001, 10000$.

Note that all the gray codes above are closed, except the ones for 9 objects. □

3. Consider the following program segment in the celebrity algorithm.

```
i := 1;
j := 2;
next := 3;
while next <= n+1 do
   if Know[i,j] then i:= next
   else j := next;
   next := next + 1;
   end;
if i = n+1 then candidate := j
else candidate := i;
```

(a) Find a loop invariant for the while loop that is sufficient to show that `candidate` will be the only possible candidate for the celebrity after the execution of the segment.
(b) Prove that the loop invariant found above is indeed a loop invariant.
Solution. (a) An appropriate loop invariant is "if k is the celebrity, then $k = i$, $k = j$, or $next \leq k \leq n$" (plus $1 \leq i \leq next$, $2 \leq j \leq next$, $3 \leq next \leq n + 2$, which is omitted for brevity).
(b) We need to show that (1) the assertion is true at the beginning of the loop and (2) given that the assertion is true and the condition of the while loop holds, the assertion will still be true after the loop body is executed.
(1) At the beginning of the loop, $i = 1$, $j = 2$, and $next = 3$. Apparently, if k is the celebrity, then $1 \leq k \leq n$ and hence $k = 1 = i$, $k = 2 = j$, or $next = 3 \leq k \leq n$.
(2) Now we assume that the assertion "if k is the celebrity, then $k = i$, $k = j$, or $next \leq k \leq n$" is true at the next iteration and the loop condition holds, i.e., $next \leq n+1$. Let i', j', and $next'$ denote respectively the values of i, j, and $next$ after the iteration. We need to show that "if k is the celebrity, then $k = i'$, $k = j'$, or $next' \leq k \leq n$". From the loop body, we deduce the following relationship:

$$i' = \begin{cases} next & \text{if } Know[i,j] \\ i & \text{otherwise} \end{cases}$$

$$j' = \begin{cases} next & \text{if } \neg Know[i,j] \\ j & \text{otherwise} \end{cases}$$

$$next' = next + 1$$

There are two cases to consider: $Know[i,j]$ and $\neg Know[i,j]$. In the first case, i cannot be the celebrity. So, the truth of "if k is the celebrity, then $k = i$, $k = j$, or $next \leq k \leq n$" implies that of "if k is the celebrity, then $k = j$, or $next \leq k \leq n$", which is equivalent to "if k is the celebrity, then $k = j$, $k = next$, or $next + 1 \leq k \leq n$". Since $i' = next$, $j' = j$ and $next' = next + 1$, it follows that "if k is the celebrity, then $k = i'$, $k = j'$, or $next' \leq k \leq n$", which concludes the first case. The second case be carried out in an analogous manner. □

4. (a) What is the result of merging the following two skylines: (1,**8**,4,**11**,9,**0**,12,**6**,18,**15**,22) and (3,**7**,13,**4**,16,**10**,24). (5 points)
   Solution. (1,**8**,4,**11**,9,**7**,13,**6**,16,**10**,18,**15**,22,**10**,24). □

   (b) Give an algorithm for merging two skylines. (10 points)
   Solution. Left as a programming exercise. □

5. In the towers of Hanoi puzzle, there are three pegs A, B, and C, with n (generalizing the original eight) disks of different sizes stacked in decreasing order on peg A. To objective is to transfer all the disks on peg A to peg B, moving one disk at a time (from one peg to one of the other two) and never having a larger disk stacked upon a smaller one.
   (a) Give an algorithm to solve the puzzle.
   Solution.

```
Algorithm Towers_Hanoi(A,B,C,n);
begin
   if n=1 then
      pop x from A and push x to B
   else
      Towers_Hanoi(A,C,B,n-1);
      pop x from A and push x to B;
      Towers_Hanoi(C,B,A,n-1);
end;
```

□

   (b) Compute the total number of moves in the algorithm.
   Solution. We count "pop x from A and push x to B" as one move. Let $T(n)$ denote the number of moves required for n disks.

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(n-1) + 1 & \text{if } n \geq 2 \end{cases}$$

Solving the equation, we get $T(n) = 2^n - 1$, for $n \geq 1$. $\square$

6. (a) Apply the partition algorithm in quicksort to the following array (assuming that the first element is chosen as the pivot).

8	2	5	11	9	12	1	15	7	3	13	4	10	16	14	6

Show the result after each exchange (swap) operation.

Solution.

8	2	5	11	9	12	1	15	7	3	13	4	10	16	14	6
8	2	5	**6**	9	12	1	15	7	3	13	4	10	16	14	**11**
8	2	5	6	**4**	12	1	15	7	3	13	**9**	10	16	14	11
8	2	5	6	4	**3**	1	15	7	**12**	13	9	10	16	14	11
8	2	5	6	4	3	1	**7**	**15**	12	13	9	10	16	14	11
7	2	5	6	4	3	1	**8**	15	12	13	9	10	16	14	11

The pair of elements swapped in each step are typeset in **boldface**.

$\square$

(b) Apply the quicksort algorithm to the above array. Show the result after each partition operation.

Solution.

8	2	5	11	9	12	1	15	7	3	13	4	10	16	14	6
7	2	5	6	4	3	1	**8**	15	12	13	9	10	16	14	11
1	2	5	6	4	3	**7**	**8**	15	12	13	9	10	16	14	11
1	*2*	5	6	4	3	**7**	**8**	15	12	13	9	10	16	14	11
1	**2**	*5*	6	4	3	**7**	**8**	15	12	13	9	10	16	14	11
1	**2**	*4*	3	**5**	6	**7**	**8**	15	12	13	9	10	16	14	11
1	**2**	3	**4**	**5**	6	**7**	**8**	*15*	12	13	9	10	16	14	11
1	**2**	3	**4**	**5**	6	**7**	**8**	*14*	12	13	9	10	11	**15**	16
1	**2**	3	**4**	**5**	6	**7**	**8**	*11*	12	13	9	10	**14**	**15**	16
1	**2**	3	**4**	**5**	6	**7**	**8**	*9*	10	**11**	13	12	**14**	**15**	16
1	**2**	3	**4**	**5**	6	**7**	**8**	**9**	10	**11**	*13*	12	**14**	**15**	16
1	**2**	3	**4**	**5**	6	**7**	**8**	**9**	10	**11**	12	**13**	**14**	**15**	16

The element that is serving as the pivot in the next partition step is typeset in *italic*. Every element that has served as a pivot is typeset in **boldface**.

$\square$

7. (a) Rearrange the following array into a heap using the bottom-up approach.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	8	5	11	9	12	1	15	7	3	13	4	10	14	6

Show the result after each element is added to the part of array that already satisfies the heap property. (5 points)

Solution.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	8	5	11	9	12	*1*	15	7	3	13	4	10	14	6
2	8	5	11	9	*12*	**14**	15	7	3	13	4	10	**1**	6
2	8	5	11	*9*	12	14	15	7	3	13	4	10	1	6
2	8	5	*11*	**13**	12	14	15	7	3	**9**	4	10	1	6
2	8	*5*	**15**	13	12	14	**11**	7	3	9	4	10	1	6
2	*8*	**14**	15	13	12	**6**	11	7	3	9	4	10	1	**5**
2	**15**	14	**11**	13	12	6	**8**	7	3	9	4	10	1	5
15	**13**	14	11	**9**	12	6	8	7	3	**2**	4	10	1	5

The element under consideration in each insertion step is typeset in *italic*. The elements that are relocated after each insertion step are typeset in **boldface**.

□

(b) Give the bottom-up heap-building algorithm (in pseudo code). (10 points)

Solution.

```
Algorithm Build_Heap(A,n);
begin
   for i := n DIV 2 downto 1 do
      parent := i;
      child1 := 2*parent;
      child2 := 2*parent + 1;
      if child2 > n then child2 := child1;
      if A[child1]>A[child2] then maxchild := child1
      else maxchild := child2;
      while maxchild<=n and A[parent]<A[maxchild] do
         swap(A[parent],A[maxchild]);
         parent := maxchild;
         child1 := 2*parent;
         child2 := 2*parent + 1;
         if child2 > n then child2 := child1;
         if A[child1]>A[child2] then maxchild := child1
         else maxchild := child2;
         end;
```

```
        end;
    end;
```

□

8. (a) Compute the *next* table (as in the KMP algorithm) for the string *ababcababdab*.

Solution.

1	2	3	4	5	6	7	8	9	10	11	12
a	b	a	b	c	a	b	a	b	d	a	b
-1	0	0	1	2	0	1	2	3	4	0	1

□

(b) Modify the KMP algorithm to find the longest prefix of string B that matches a substring of A, assuming the *next* table for string B is given.

Solution.

```
Algorithm Longest_Prefix(A,n,B,m);
begin
   i := 1;
   j := 1;
   Start := 0;
   MaxLength := 0;
   while MaxLength<m and i<=n do
      if A[i] = B[j] then
         i := i + 1;
         j := j + 1
      else
         if (j-1)>MaxLength then
            MaxLength := j - 1;
            Start := i - j;
         j := next[j] + 1;
         if j=0 then
            j := 1;
            i := i + 1;
      if j=m+1 then
         MaxLength := m;
         Start := i - m;
      end;
end;
```

□

9. Given two strings *baa* and *bcba*, compute the minimal cost matrix $C[0..3, 0..4]$ for changing the first string character by character to the second one.

Solution.

	0	1	2	3	4
0	0	1	2	3	4
1	1	0	1	2	3
2	2	1	1	2	2
3	3	2	2	2	2

$\square$