Data Structures & Algorithms (CS F211) Mid Sem Exam

There are 11 questions in all and total marks is 40. For questions 1 to 10: please write your answers at one place in sequential order. There is no need to give explanations. For question 11: please show all steps in computations and proofs. This is a **closed book exam**. Time: 90 minutes.

1. Convert the following infix expression into postfix: [2]
   $A + (((B - C) * (D - E) + F)/G)\$(H - J)$

2. Draw an almost complete binary tree having 10 nodes. [2]

3. Postorder traversal of a binary tree is $IEJFCGKLHDBA$, and its inorder traversal is $EICFJBGDKHLA$. Draw the binary tree. [2]

4. Use Build-Max-Heap to convert the array $< 5, 3, 17, 10, 84, 19, 6, 22, 9 >$ into a max heap. [2]

5. The elements of the array $< 2, 252, 401, 398, 330, 344, 397, 363 >$ are inserted into an empty binary search tree in order from left to right. Draw the final binary search tree. [2]

6. By selecting the last element of the array as the pivot, apply the first partitioning according to the Quicksort algorithm for the array: [2]
   $< 14, 17, 13, 15, 19, 10, 3, 16, 9, 12 >$

7. Let $A[1, ..., n]$ be an array storing a bit (1 or 0) at each location, and $f(m)$ be a function whose time complexity is $\Theta(m)$. Find the time complexity of the following program fragment written in a C like language: [2]

```
counter = 0;
for(i = 1; i <= n; i++)
{
  if(A[i] == 1)
  {
    counter++;
  }
  else
  {
    f(counter);
    counter = 0;
  }
}
```

8. Find the time complexity of the following C function (assuming $n > 0$): [2]

```
int recursive(int n)
{
  if(n == 1)
  {
    return (1);
  }
  else
  {
    return (recursive(n - 1) + recursive(n - 1));
  }
}
```

9. Solve the recurrence relation: [2]
   $T(1) = 1$
   $T(n) = 2T(n-1) + n, (n \geq 2)$

10. Find the time complexity for merging two sorted arrays of size m and n. [2]

11. Suppose that we are given a key k to search for in a hash table with positions $0, 1, ..., m-1$, and suppose that we have a hash function h' mapping the key space into the set $\{0, 1, ..., m-1\}$. The search scheme is as follows:

    1. Compute the value $j = h'(k)$, and set $i = 0$.
    2. Probe in position j for the desired key k. If you find it, or if this position is empty, terminate the search.
    3. Set $i = i + 1$. If i now equals m, the table is full, so terminate the search. Otherwise, set $j = (i + j) \bmod m$, and return to step 2.

    Assume that m is a power of 2.

    (a) Show that this scheme is an instance of the general "quadratic probing" scheme by exhibiting the appropriate constants c_1 and c_2 for the equation $h(k, i) = (h'(k) + c_1 i + c_2 i^2) \bmod m$. [8]

    (b) Prove that this algorithm examines every table position in the worst case. [12]