**Data Structures** **Test I Part II** **Max Marks: 15**

1. Prove that $(\log n)^2 = o(n)$. [2]

   **Soln:** By L'Hospital's rule, $\lim_{n\to\infty} \frac{\log n}{n^2} = \lim_{n\to\infty} \frac{\frac{1}{n}}{2n} = 0$.

2. Suppose you are merging an $m$ element array with an $n$ element array. What is the minimum number of comparisons required (in the best case)? Justify your answer. [2]

   **Soln:** When all elements of the smaller array is smaller than all elements of the larger array, $\min(m, n)$ comparisions suffice.

3. In the version of quick sort eliminating tail recursion, Assume that we always invoke recursively the partition of smaller size and do the larger part iteratively. Let $c$ be constant indicating the amount of stack memory required for each recursive all. Let $S(n)$ denote the worst case stack memory required for sorting an array of $n$ elements. Write down a recurrence for estimating $S(n)$. Justify your answer and solve the recurrence. [2]

   **Soln:** If $T(n)$ is the stack size for an array of size $n$ and if each call takes $c$ space, then since each recursive call reduces the value of $n$ by at least to half, we have $T(n) = T(\frac{n}{2}) + c = \theta(\log n)$.

4. Let $t$ be a pointer to the $root$ of a linked list defined over the following node structure: [4]

```
typedef struct node{
     int data;
     struct node *next;
};
```

   Eliminate recursion using a **single** while loop. (Assume functions push(struct node *t), struct node* pop() and int stackempty() in the standard manner) [ Answer on the reverse side]

```
void test(struct node * t) {
   if (t==NULL) return;
   else {
       test(t->next);
       print(t->data);
   }
   return;
}
```

   *Soln:* This question is deliberately left unsolved.

5. Let $A$ be an $n$ element array. We want $A$ to store a randomly generated permutation of the set $\{1, 2, .., n\}$. Assume that we have a function $Pick()$ that picks an element from the set $\{1, 2, .., n\}$ uniformly at random. Consider the following algorithm: [Answer on the reverse side] [5]

```
A[1] = Pick()
for i = 2 to n do
    L :  x = Pick()
         if x is equal to one among A[1], A[2],.., A[i-1], goto L
         A[i] = x;
endfor
```

   Compute the expected number of times the function $Pick()$ will be invoked before the algorithm completes execution.

   **Soln:** For each $i$ iteration, probability that each call to pick picks an element different from the ones chosen previously is $\frac{n-i+1}{n}$. The expected number of calls to $Pick()$ for each $i$ is given by the mean of the geometric distribution, $\frac{n}{n-i+1}$. Summing over all values of $i$ and adding the first call to $Pick()$ outside the loop, the total cost if $1 + \sum_{i=2}^{n} \frac{n}{n-i+1} = nH_n = \theta(n \log n)$.