Computer Science I
Final Back Paper - December 29th, 2011

Max Marks: 50 **Time: 3 Hrs**

Answer ANY 4 out of questions 1-6 (*Algorithms*) and ANY 2 from questions 7-9 (*Programming*)

---

1. Consider an array of **11** numbers **A=[2, 15, 4, 9, 26, 33, 1, 13, 20, 17, 28]**. Trace the working of quicksort on this array, to sort the array in the increasing order of the numbers. Show the arrangement of the numbers in **A** after each major step in the algorithm. **Marks: 9**

2. Repeat problem 1 with Heapsort. **Marks: 9**

3. Use the definition of the O-notation to show that $T(n) = n^{O(1)}$ if and only if there exists a constant $k>0$ such that $T(n)=O(n^k)$. **Marks: 9**

4. What is the optimal Huffman code for the following set of frequencies based on the first eight Fibonacci numbers - **a:1**, **b:1**, **c:2**, **d:3**, **e:5**, **f:8**, **g:13**, **h:21**. Generalize this to find the optimal code for **n** characters whose frequencies are the first **n** Fibonacci numbers. Fibonacci numbers are defined as: $F_1=1, F_2=1, F_3=2, \ldots, F_i = F_{i-1} + F_{i-2}$. **Marks: 9**

5.
   a. A complete binary tree is one where every non-leaf node has exactly two children. The depth of such a tree is the length of the longest path from the root of the tree to a leaf. Find the maximum and minimum number of nodes that a complete binary tree of depth **d** can have and illustrate how the trees would look. **Marks: 5**

   b. Find the maximum and minimum number of leaves in a complete binary tree. Correlate the trees for which this would happen with those in part (a). **Marks: 2**

   c. When is the worst case running time of Quicksort is optimal? (a) when the pivot element is at a fixed position, (ii) when the pivot element is the median element, (iii) when the pivot element is a random element. Justify your answer. **Marks: 2**

6.
   a. Solve the recurrence relation $T(n) = 2T(n/3) + O(n)$. **Marks: 4**

   b. Give a divide-and-conquer algorithm that runs in time $O(\log n)$ to find out if there is an index **i** for which $A[i]=i$ in a sorted array of distinct integers $A[1...n]$. **Marks: 5**

7.

a. Define a class **Rational**, that represents a rational number with the numerator and denominator as integers. Provide 2 constructors for the class, one that takes another rational as an argument, another constructor that takes two integers as arguments while ensuring that a single integer argument can also be given (integers themselves are obviously also rationals) Also write the code for overloaded + and * operators for the **Rational** class. Also define an invert method that finds the reciprocal of the rational number. **Marks: 3**

b. Define a **reduce** method for the **Rational** class that reduces the rational to its canonical form by dividing both the numerator and denominator by their GCD. Define a function as an iterative version of the following GCD algorithm and use it to define the **reduce** method. **Marks: 4**

Euclid's GCD Algorithm: Take two integers **m** and **n**, **m>=n**. If **n=0** then **GCD(m,n)** is **m**. Otherwise let **x** be **(m mod n)**, and replace **m** and **n** by **n** and **x** respectively and repeat – it can be shown that **GCD(m,n) = GCD(n,x)**.

8.

a. Write a C++ function to check if a given integer is a palindrome (reads the same when read from left-to-right and right-to-left) when written in its decimal form. Fill in the code for the function **isPalindrome** below.

```
#include <iostream>

using namespace std;

bool isPalindrome(int n) {
        ... // fill this in
}

int main() {
        long n;
        cin>>n;
        cout<< isPalindrome(n);
        return 0;
}
```

Marks: 4

b. Write a C++ program to pairwise-invert any given string of characters. For instance given a string "**abcdefgh**", the program should output "**badcfehg**" and for "**x1y2z3s**" it outputs "**1x2y3zs**". Note that for odd-length strings, the last character is left as is. Fill in the body of the "**pairwiseInvert**" function as in the code below.

```
#include <iostream>
#nclude <string>
using namespace std;

string pairwiseInvert (string& s) {
        ... // fill this in
}

int main() {
        string s;
        cin>>s;
        cout<< pairwiseInvert (s);
        return 0;
}
```

Assume that the **string** class supports a [ ] operator to get the i^{th} character of the string and a **length** method returning the number of characters in the string. **Marks: 3**

9.

a. Create a C++ function to check if a given linked list is a circularly linked list. Fill in the body of the **isCircular** function in the code below. **Marks: 3**

```
#include <iostream>
#include "LinkNode.h"
#include "LinkedList.h"
using namespace std;

bool isCircular (LinkedList& l) {
        ... // fill this in
}

int main() {
        LinkedList list;
        /* some code to populate the list */
        cout<< isCircular(list);
        return 0;
}
```

Assume the **LinkedList** class supports the following methods:

void LinkedList::reset(void); *// moves the current pointer to the head of the list*

int LinkedList::length(void); *// returns the number of elements in the list*

LinkNode* LinkedList::currentNode(void); *// returns a pointer to the current node*

void LinkedList::next(void); *// moves the current pointer to the next (null if it has reached the end of the list) node*

void LinkedList::setNext(LinkNode * p); *// sets the next pointer of current node to* ***p***

b. Create a C++ function to reverse the linked list in the code above. **Marks: 4**