Name:_____ **Entry No:**_____

COL106: Data Structures. I semester, 2015-16.
Minor I
2:30 PM to 3:30 PM, 1st September 2015.

Question	1 (4 marks)	2 (6 marks)	3 (6 marks)	4 (4 marks)	Total (20 marks)
Marks					

Write your answers on the **printed question paper** in the space provided. **ROUGH SHEETS WILL NOT BE COLLECTED.**

Q1. (Time Complexity and O-notation. Total marks = 4). Marks:

Suppose we have some already defined functions `g(n)` and `h(n)`, consider the following function definition of the function `f(n)`.

```
1: void f (int n)
2: {
3:   c = 1;
4:   for i = 1 to n
5:  {
6:     print g(i);
7:     if (i = c)
8:       then
9:       {
10:           print h(i);
11:           c = 2*c;
12:       }
13:   }
14: }
```

In the following we will assume that each assignment statement (e.g. `c = 1`) takes 1 unit of time, each time the `for` statement is run it takes 2 units of time, each a print statement is executed it takes 1 unit of time, and each multiplication takes 1 unit of time.

In each of the following questions you have to calculate the *exact* number of time units it takes for `f(n)` to run, and *also* write the time complexity in simplified big-Oh notation, e.g., if the number of time units turns out to be $3n^3 + 2\log n$, you must give the final answer as $O(n^3)$.

Q1.1. (1.5 marks) Calculate the number of time units and time complexity of `f(n)`, if `g(i)` takes i units of time and `h(i)` takes 1 unit of time.

**** Solution ****

The total time taken is

1 for line #3 + $2n$ for line #4 + n + $\sum_{i=1}^{n} T(g(i))$ for line #6 (1 for the print statement and $T(g(i))$ for the execution of $g(i)$.

The if statement returns true whenever i is a power of 2 which happens $\lfloor \log n \rfloor$ times. Hence the print statement on line #10 runs $\lfloor \log n \rfloor$ times, and so does the multiplication in line #11. The time taken by the executions of $h(i)$ can be written as $\sum_{j=1}^{\lfloor \log n \rfloor} T(2^j)$.

Hence the total time is

$$1 + 3n + \sum_{i=1}^{n} T(g(i)) + 2\lfloor \log n \rfloor + \sum_{j=1}^{\lfloor \log n \rfloor} T(2^j).$$

In Q1.1, putting $T(g(i)) = i$ and $T(h(i)) = 1$, this reduces to

$$1 + 3n + \sum_{i=1}^{n} T(g(i)) + 3\lfloor \log n \rfloor,$$

which is

$$1 + 3n + \frac{n(n+1)}{2} + 3\lfloor \log n \rfloor.$$

The final answer is $O(n^2)$.

Marks breakup. 1 mark for the exact steps expression. If the $\lfloor$ and $\rfloor$ are not there don't deduct marks. Some students asked if the comparison in line 7 should be counted. Since I had not written a number of steps for that, we will consider it okay if they have included it (using a value of 1 step per comparison) or if they have left it out. If they have included it then the expression above should have $2n$ rather than just n. Either way they get full marks. To mark the exact number of steps leniently, we will give 1 mark if there is one $O(1)$ term, one $O(n)$ term, one $O(n^2)$ term and one $O(n \log n)$ term.

0.5 marks for the $O(n^2)$. No partial credit for either part, i.e., either 1 or 0 for the exact number of step and either 0.5 or 0 for the big-Oh part.

Q1.2. (2.5 marks) Calculate the number of time units and time complexity of `f(n)`, if `g(i)` takes $\log i$ units of time and `h(i)` takes i units of time.

**** Solution ****

Using the expression above we get

$$1 + n + 2\lfloor \log n \rfloor + \sum_{i=1}^{n} \log i + \sum_{j=1}^{\lfloor \log n \rfloor} 2^j.$$

Here $\sum_{i=1}^{n} \log i$ is $O(n \log n)$ and $\sum_{j=1}^{\lfloor \log n \rfloor} 2^j$ is $O(n)$ and so the final answer is $O(n \log n)$.

Marks breakup. 1 mark for the expression, 1.5 marks for simplifying it into big-Oh notation correctly. No partial credit for either part. Remember the point about the running time of the comparison in line 7.

Q2. (Linked Lists. Total marks = 6). Marks:

Consider the following function that takes two linked lists as input:

```
list lf (list a, list b)
{
   temp1 = a;

   if b = null
```

```
     then
       throw exception Ex;
     else
       temp2 = b;

   temp3 = null;

   while (temp1 not equal to null)
   {
       if (temp2 = null)
          then
          {
            x = new node;
            x.next = temp3;
            temp3 = x;
            temp2 = b;
          }
       temp2 = temp2.next;
       temp1 = temp1.next;
   }

   return temp3;
}
```

Name:_____ **Entry No:**_____

Q2.1. (3.5 marks) Suppose we call `lf` with the list `a` having 7 nodes and `b` having 3 nodes, show the state of the three lists (`a, b, temp3`) and the positions of the pointers `temp1, temp2` at the end of each iteration of the while loop. Note that the function `lf` does not do anything to the data in the nodes so you can draw the list with the space for data left empty.

**** Solution ****

Marks. There should be 7 steps in all. Give 0.5 marks for each step. If there are fewer steps, give marks considering them as the first 5 steps. If there are more steps, deduct 0.5 marks for each extra step.

Q2.2. (1 mark) Explain in words what the function `lf` does.

**** Solution ****

Given a linked list `a` with m nodes and a linked list `b` with n nodes where $n > 0$, `lf` returns a linked list with $m - 1/n$ nodes where the division is integer division.

Marks. If $n > 0$ is not mentioned please deduct 0.5 marks. Instead of $m-1/n$, to be lenient we can give a full mark if they have written m/n.

Q2.3. (1.5 marks) Assume that the only operation that takes 1 unit of time is the `.next` operation (e.g. `temp1.next`) and all other operations take 0 units of time. Calculate the amount of time taken by `lf` if the size of `a` is m and the size of `b` is n. Your answer must be in terms of m and n. Calculate the exact number of steps and also give the answer in terms of big-oh notation.

**** Solution ****

The exact number of steps is $2m+m/n$ since there are two `.next` operations in each iteration of the loop and one `.next` operation is run every time `temp2` comes to the end of `b`. Hence running time is $O(m)$.

Marks breakup. 1 mark for exact number of steps and 0.5 marks for big-Oh notation. No partial credit for either.

Q3. (Queues. 6 marks). Marks:

A queue contains three types of characters R (red), W (white) and B (blue) in a jumbled up (assorted) fashion. The total number of characters is say n and n is known to you. Write an algorithm to arrange these characters in the queue such that all the R's come first, the W's come next, and B's come last in the queue. You may use one additional queue and one or two constant space variables if required. Your algorithm is **not** allowed to create or destroy any character of the input, it can only **rearrange** them. **Please write your algorithm in plain English giving exact steps. *No code or pseudocode. If you write code or pseudocode you automatically get 0*.**

**** Solution ****

There are many ways of doing this. The simplest is as follows:

- Since you know n, dequeue from Q1 n times. Every time you get an R enqueue it in Q2. Enqueue the others back in Q1.
- Count the number of Rs you enqueued in Q2. Let's say the number if k.
- Dequeue from Q1 $n - k$ times. Enqueue the Ws into Q2 and the Bs back into Q1.
- Finally dequeue from Q1 (which now contains only Bs) and enqueue into Q2 till Q1 is empty.

To this simple solution you may encounter some variants, e.g., someone may forget that n is given in advance and may first dequeue Q1 completely and enqueue everything into Q2 just

to find out the value of n. Another possibility is enqueuing Rs back into Q1 and W and B into Q2 in the first step. All these are basically correct.

Other solutions are also possible including one (I think) where we do not use the number n or $n - k$, so if you encounter a solution which doesn't resemble the one above, please read it carefully.

Marks. If the basic outline of the solution is there then the student should get at least 4 marks. If it doesn't appear to be a basic understanding of the nature of the solution the student should get 3 or less. You will have to read this carefully to figure out how close the answer is to the correct one.

Q4. (Trees. Total marks = 5) Marks:

Suppose we are given a java implementation which has a `Tree` class and a `Node` class. The `Tree` class has the following methods available:

Omitted to save space in the solutions document.

Q4.1. (1.5 marks): Given a tree `T` explain in words what the function `f1(T)` outputs?

**** Solution ****

`f1(T)` outputs the sum of all the non-negative (positive) values in the tree.

Marks. No partial credit. It should clearly state that the sum is of all *non-neagtive* or *positive* entries/data values stored in the tree.

Q4.2. (2.5 marks): Given a tree `T` explain in words what the function `f2(T)` outputs?

**** Solution ****

`f2(T)` outputs 1 if the tree is a binary tree i.e. if all the nodes have at most two children, and 0 otherwise.

Marks. You will have to read this answer carefully to understand whether the student understands what is happening here. Some partial credit may be given.
