

Name: SankalpEntry No: 2014PH10822

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	4	9.5	0	4	13.5

288

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: 4

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

```
void f (int n)
{
    c = 1;
    for i = 1 to n
    {
        print g(i);
        if (i = c)
            then
            {
                print h(i);
                c = 2*c;
            }
    }
}
```

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.

time units

assuming each comparison takes 1 unit of time

$$\text{of } f(n) = 1 + [i+4]n + 3 \log_2 n$$

$$1 + 4n + \frac{n(n+1)}{2} + 3 \log_2 n = \frac{1}{2}n^2 + \frac{9}{2}n + 3 \log_2 n + 1$$

Time complexity

 $O(n^2)$

1.5

Name: _____

Entry No: _____

Name: _____

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.

[.] \rightarrow g.i.ftime units taken by $f(n)$

$$= 1 + 4n + \sum_{i=1}^n \log i + \sum_{i=1}^{\lfloor \log_2 n \rfloor} (2+i) \cdot 2 \log_2 n$$

$$= 1 + 4n + \log(n!) + 2 \log_2 n + \frac{\lfloor \log_2 n \rfloor (\lfloor \log_2 n \rfloor + 1)}{2}$$

Time complexity $O(\log(n!))$

Marks: 5.5

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

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;
    }

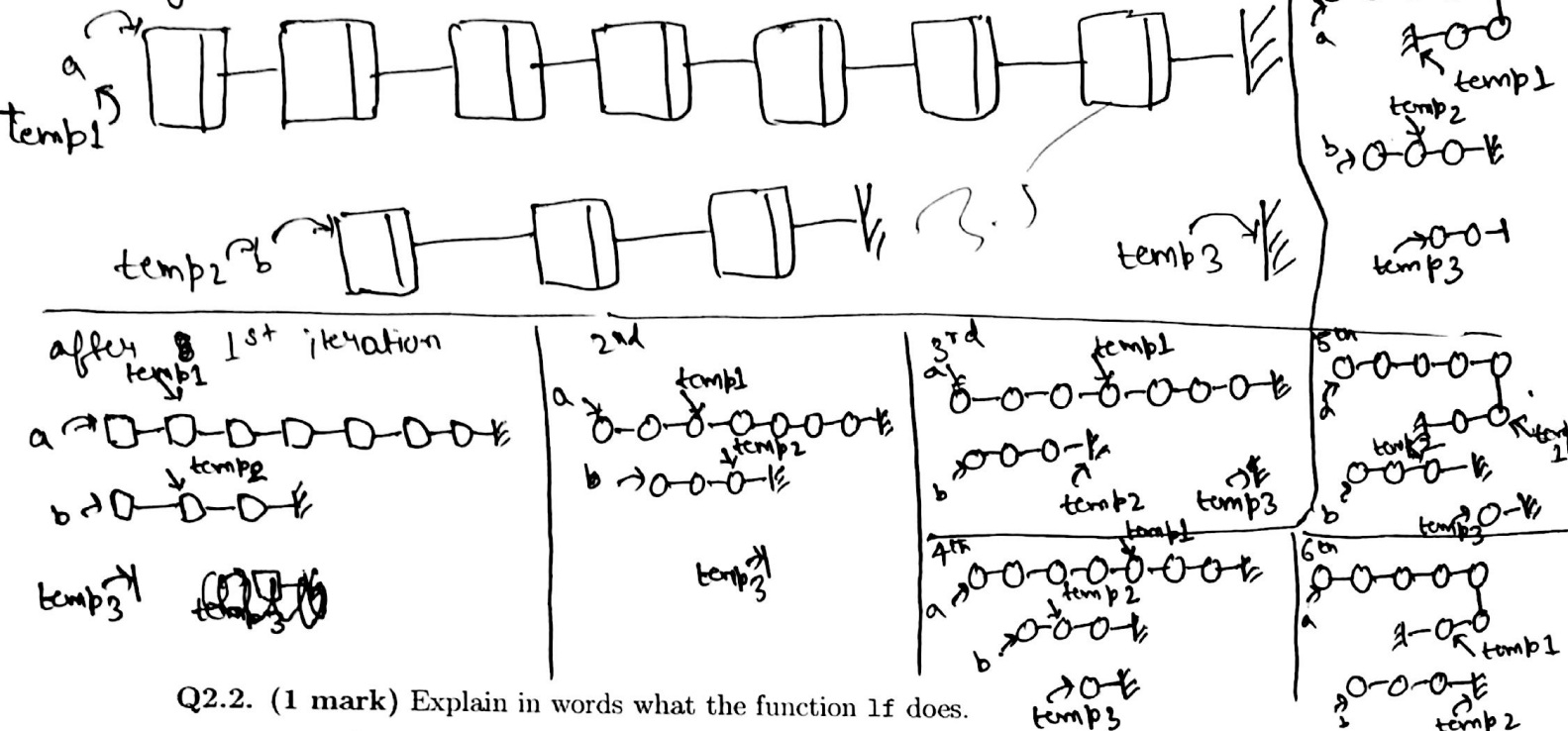
    return temp3;
}
```

Name: Samkalp

Entry No: 2014PH10822

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.

Initially



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

The function `lf` returns a linked list with no. of nodes equal to ~~no. of~~ least integer greatest integer before the value of no. of nodes of `a` divided by no. of nodes of `b`.

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.

$$\begin{aligned} \text{time taken by I} &= m + m + \left\lceil \frac{m}{n} \right\rceil \\ &= 2m + \left\lceil \frac{m}{n} \right\rceil \end{aligned}$$

$$\text{time complexity} = O(m)$$

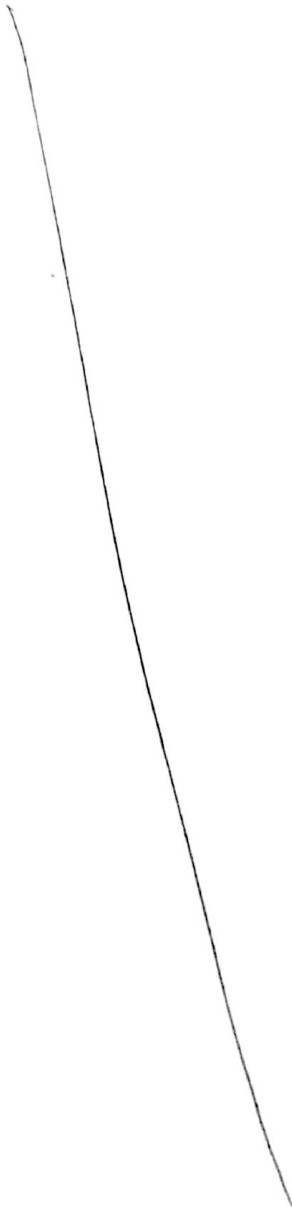
Name: _____

Entry No: _____

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.**



Name: Sankalp

Entry No: 2014PH10822

Q4. (Trees. Total marks = 5)

Marks: 4

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

- `public Node root()`: This returns the root of the tree.
- `public Boolean isEmpty()`: This returns 1 if the tree is empty, 0 otherwise.

The Node class has the following methods available:

- `public int data()`: This returns the integer data value stored in the node.
- `public int no_children()`: This returns the number of children of the node.
- `public Node child(int i)`: This returns the Node of the i th child, returning null if there are less than i children.
- `public Tree subtree(int i)`: This returns the Tree rooted at the i th child, returning an empty tree if there are less than i children.

Consider the following functions:

```
public int f1 (Tree T) {  
    if (T.isEmpty()) {  
        return 0;  
    }  
    else {  
        Node curr = T.root();  
        int temp = curr.data();  
        if (temp < 0) { temp = 0;}  
        int i = 0;  
  
        while (i < curr.no_children()) {  
            temp = temp + f1(curr.subtree(i));  
            i++;  
        }  
        return temp;  
    }  
}
```

```
public int f2 (Tree T) {  
    if (T.isEmpty()) {  
        return 1;  
    }  
    else {  
        Node curr = T.root();  
        if (curr.no_children() > 2) {
```

Name: _____

Entry No: _____

```
    return 0;
}
else {
    int temp = 1;
    int i = 0;

    while (i < curr.no_children()) {
        if (temp > f2(curr.subtree(i)) {
            temp = 0;
        }
        else
            i++;
    }
    return temp;
}
```

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

f1(T) gives the sum of all the non -ve values stored in that tree. (each and every node/^{subtree} is considered and the value is added to a counter if it is non -ve, finally the counter is returned)

1.5

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

returns zero if not a binary tree/subtree
For binary tree/subtree returns 1

2.5