Solution to CSE 250 Midterm Exam 2

Fall 2014
Time: 50 minutes.

Total points: 100	There are 4 questions.

Please use the space provided for each question, and the back of the page if you need to. Please do not use any extra paper. The space given per question is **a lot** more than sufficient to answer the question. Please be brief. Longer answers do not get more points!

- **No electronic devices of any kind. You can open your textbook and notes**
- **Please leave your UB ID card on the table**
- **This booklet must not be torn or mutilated in any way and must not be taken from the exam room**
- **Please stop writing when you are told to do so. We will not accept your submission otherwise.**

Your name:	_____
UBIT ID:	_____
CSE 250 Index Number:	_____

The rest of this page is for official use only. Do not write on the page beyond this point.

Problem Number	Score obtained
name, UBIT Name, Index Number (5 max)	
Problem 1 (20 max)	
Problem 2 (15 max)	
Problem 3 (30 max)	
Problem 4 (30 max)	
Total Score: (100 max)	

Problem 1 (20 points)**.** Order the following functions in increasing order of asymptotic growth rate. You don't have to explain how you get the order.

$$n\log^3 n,\ \ \frac{3^{2n}}{n^3},\ \ \frac{n\sqrt{n}}{\log^2 n},\ \ n^3,\ \ \log^4 n,\ \ n^{1.5},\ \ 2^{3n},$$

Answer.

$$\log^4 n,\ \ n\log^3 n,\ \ \frac{n\sqrt{n}}{\log^2 n},\ \ n^{1.5},\ \ n^3,\ \ 2^{3n},\ \ \frac{3^{2n}}{n^3},$$

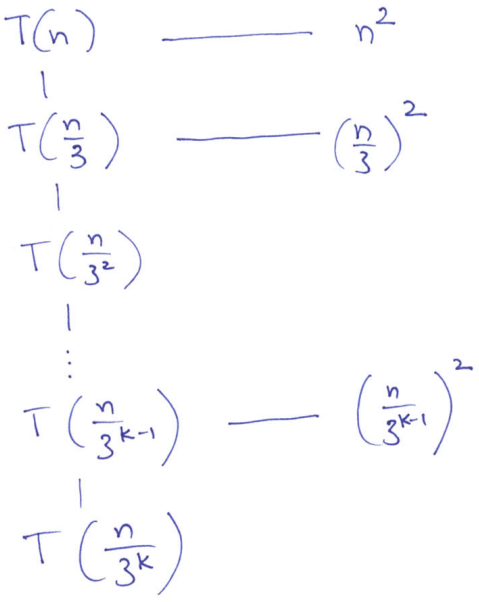
□

Problem 2 (15 points)**.** Use the recurrence tree method to solve the following recurrence relation

$$T(n) = T(n/3) + n^2$$

As usual, you can assume $T(k) = O(1)$ for $k \leq 10$. Show your work: **draw the tree and do the summation**.

Answer. The recurrence tree is as follows.



Thus, we have

$$\begin{aligned} T(n) &= T(n/3^k) + n^2 + (n/3)^2 + (n/3^2)^2 + \cdots + (n/3^{k-1})^2 \\ &= T(n/3^k) + n^2\left(1 + 1/9 + 1/9^2 + \cdots + 1/9^{k-1}\right) \\ &= T(n/3^k) + n^2\left(\frac{1 - 1/9^k}{1 - 1/9}\right). \end{aligned}$$

By setting $k = \log_3 n$, we have $n/3^k = 1$ and $9^k = 9^{\log_3(n)} = n^{\log_3 9} = n^2$. Thus

$$T(n) = T(1) + n^2\frac{1 - 1/n^2}{8/9} = T(1) + 9n^2/8 - 9/8 = \Theta(n^2).$$

□

Problem 3 (30 points)**.** Let `Node` be the following structure

```
struct Node {
    int key;
    Node* next;
};
```

Write **a recursive version** and an **iterative version** of a C++ function that takes a pointer to `Node` which is the head of a `NULL`-terminated singly linked list and returns an integer which is the following alternating sum: first key *minus* second key *plus* third key *minus* fourth key, and so forth. For example,

- input : $a.4 \to b.2 \to c.5 \to d.2$, output : 5, because $5 = 4 - 2 + 5 - 2$
- input : $a.4 \to b.2 \to c.5$, output : 7, because $7 = 4 - 2 + 5$
- input : EMPTY – output : 0

```
int iterative_alt_sum(Node* head)
{
    int sum=0;
    size_t i=0;
    for (; head != NULL; head = head->next) {
        if (i % 2 == 0)
            sum += head->key;
        else
            sum -= head->key;
        i++;
    }
    return sum;
}

int recursive_alt_sum(Node* head)
{
    if (head == NULL)
        return 0;
    else
        return head->key - recursive_alt_sum(head->next);
}
```

Problem 4 (30 points)**.** Let `Node` be the following structure

```
struct Node
{
    int key;
    Node* next;
}
```

Write **a recursive C++ function** that takes a pointer to `Node` which is the head of a `NULL`-terminated singly linked list, and a key x. The function removes all nodes with key $= x$ and returns the pointer to the head of the resulting linked list. **Only pointer manipulation is allowed.** (This is essentially the same function required in Assignment 6, but a recursive solution is asked.) Remember to free up memory of the removed nodes too. For example,

- Input: $a.4 \to b.5 \to c.4 \to d.3$, $x = 4$; output: pointer to b, where the resulting list is $b.5 \to d.3$.

```
Node* recursive_remove(Node* head, int x)
{
    if (head == NULL)
        return NULL;
    if (head->key == x) {
        Node* tmp = head->next;
        delete head;
        return recursive_remove(tmp);
    } else {
        head->next = recursive_remove(head->next);
        return head;
    }
}
```