big o notation

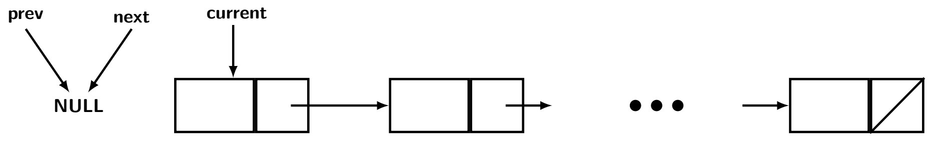
CS102: QUIZ - 1, 25th January 2020

Full time for this Quiz is 45 mts & Full marks 30.
Mention your name and roll number in each page of the paper in the space provided.
Answer the questions in the spaces provided on the question sheets.
No partial credit without appropriate (be brief) explanation

Name:______________________________

Roll No:______________________________

1. (5 points) The initial values of pointers current, next and prev with respect to a linked list is given in the figure below.



Using the initial configuration of the pointers, a function for the reversal of a linked is provided below. It does not use any extra variable. However, the parts of the code are blanked out. Your task is to supply appropriate code snippets to replace the blanks so that it works as state.

```
void reverse() {
        NODE *current = ___1___;
        NODE *prev = ___2___;
        NODE *next = ___3___;



        while (current != ___4___) {
            // Store next of current
            next = ___5___;
            // Reverse current
            ___6___ = ___7___;

            // Move pointers one position ahead.
            ___8___ = current;
            current = ___9___;
        }
        head =___10___;   // Set head
}
```

The complete solution is:

```
void reverse() {
        // Initialize current, previous and
        // next pointers
        Node* current = head;
        Node *prev = NULL;
        Node *next = NULL;

        while (current != NULL) {
            // Store next
            next = current->next;

            // Reverse current node's pointer
            current->next = prev;
            // Move pointers one position ahead.
            prev = current;
            current = next;
        }
        head = prev;
}
```

[0.5 for each currect answer]

2. (5 points) Suppose $T_1(n) = O(f(n))$ and $T_2(n) = O(f(n))$ which of the following is true? If it is not true, give appropriate explanation with a counter example in each cse.
   (A) $T_1(n) + T_2(n) = O(f(n))$.
   (B) $\frac{T_1(n)}{T_2(n)} = O(1)$.
   (C) $T_1(n) = O(T_2(n))$

Answer: [+2 points] ONLY (A) is correct as $\max f(n), f(n) = f(n)$.

[+2 points] (B) is incorrect. To see it consider $T_1(n) = n^2$ and $T_2(n) = n$, and $f(n) = n^2$. Then

$$\begin{aligned}\lim_{n\to\infty} \frac{T_1(n)}{T_2(n)} &= \lim_{n\to\infty} \frac{n^2}{n} \\ &= \infty\end{aligned}$$

[+1 points] (C) is also incorrect. The counter example provided in (B) also works in this case

3. (5 points) A quadratic algorithm with processing time $T(n) = cn^2$ spends $T(N)$ seconds for processing N data items. How much time will be spent in seconds for processing $n = 5000$ data items, assuming that $N = 100$ and $T(N) = 2$ms?

Answer: [+5 points, -1 if only the computation wrong] The constant factor $c = \frac{T(N)}{N^2}$. Therefore, $\frac{T(N)}{N^2}n^2 = T(N)\frac{n^2}{N^2} = \frac{n^2}{10^4} \times 2ms$, and $T(5000) = 5$s.

4. (5 points) Use big-Oh notation to find the tightest upperbound for running time of the following functions used for computing m^n.

```
int pow(int m, int n) {
    int result = 1;
    int k = m;
    int i = n;
    while (i > 0) {
        if (i % 2 == 1)
            result *= k;
        k *= k;
        i /= 2;
    }
    return result;
}
```

Answer: The running time depends on complexity of while loop. While loop is executed until $i > 0$. Since initial value of $i = n$ and every time i is halved, i becomes 0 after $1 + \log n$ times. Assuming each arithmetic operation take 1 unit time, the tightest upper bound for the code is O($\log n$).

[Full mark for the answers with explanation that i is halved every iteration of the loop]

5. (5 points) A template for proof that number of leaves in a strictly binary tree is 1 more than number the number internal nodes is given below. Study the proof carefully and replace each blank appropriately to create a valid proof.

Proof Template

Let the number of leaves and the number of internal nodes in strictly binary tree of height h be L_h and I_h respectively.

Induction basis: A strictly binary tree of height 1 has `__1__` leaves and `__2__` internal node.

big o notation Induction hypothesis: Assume that the results holds for any strictly binary tree of height $\leq h-1$.

Induction step: Now consider a strictly binary tree T_h of height h. Remove `__3__` node from T_h. It splits T_h into two subtrees, one left subtree and one right subtree with height of each subtree being at most `__4__`.

(i) Let the number leaves and the number of internal nodes in left subtree be L_1 and I_1 respectively.

(ii) Similarly, let the number leaves and the number of internal nodes in right subtree be L_2 and I_2 respectively.

Applying induction hypothesis to the subtrees, we have,

$L_1 =$ `__5__` and `__6__` $= I_2 + 1$.

But, the total number of internal nodes in T_h is

$I_h = I_1 + I_2 +$ `__7__`

Thereforbig o notation e, $L_h =$ `__8__` $+$ `__9__` $+$ `__10__`

Answer:

Blank No.	**Replacement text/symbol**
1.	2
2.	1
3.	the root
4.	$h-1$
5.	I_1
6.	L_2
7.	1
8.	I_1
9.	I_2
10.	1

6. (5 points) Place a tick mark in the cell (i, j) of the table below if the two conditions represented by row i and col j can occur simultaneously.

Answer:

	preorder(x) < preorder(y)	postorder(x) < postorder(y)	inorder(x) < inorder(y)
x is to the left of y	x		x
x is to the right of y		x	x
x is a proper ancesor of y	x		x
x is a proper descendant of y		x	