CMPSC 24 Spring 2018 Midterm-II Answer Sheet

Name: [redacted] Seat no. [redacted] Perm no. : [redacted]

Student to your left: [redacted] Student to your right [redacted]

Please ask all questions in writing using the scratch sheet.

Part 1: Q1.[10 pts]: Implement `insertFront()` and derive its Big-O running time

```
void LinkedList::insertFront (int value){
    LinkedList::Node * n = new LinkedList::Node(value);
    n->next = head;
    head = n;
    if (tail == NULL) //empty list
        tail = n;

}
```

] All operations are constant time

Big-O: O(1)

CMPSC 24 Spring 2018 Midterm-II Answer Sheet

Name: [redacted] Seat no. ______ Perm no. :____________

Student to your left:____________ Student to your right ____________

Part 1 Q2 [6 pts]: Implement `destructor` of the `LinkedList` and `Node` classes and derive the Big-O running time of ~LinkedList()

```
LinkedList:: ~LinkedList(){
    delete head;
}

LinkedList:: ~Node(){
    delete next;
}
```

] N recursive calls

$O(1) + O(N) = O(N)$

$\boxed{\text{Big-O: } O(N)}$

CMPSC 24 Spring 2018 Midterm-II Answer Sheet

Name: [redacted]____________ Seat no. _______ Perm no. :_______________

Student to your left:______________________ Student to your right ____________________

Q3.Big O running time of append( ). You may annotate the running time of blocks of code in your derivation.

(a) Implementation #1

```
LinkedList::append(int value){

      if(head == 0){
            head = new Node(value);     ] O(1)
            tail = head;
      } else{
            tail->next = new Node(value);     ] O(1)
            tail = tail->next;
      }

}
```

Overall Big-O running time for implementation #1 O(1)

(b) Implementation #2

```
LinkedList::append(int value){
      if(head == 0){
            head = new Node(value); 1      O(1)
            tail = head; 1
      } else{
           Node* tmp = head; 1
           while(tmp->next)   N           ] O(N)
              tmp=tmp->next;
            tmp->next = new Node(value);
            tail = tmp;                     worst case O(N)
      }
}
```

Overall Big-O running time for implementation #1 O(N)

Part 2: More running time analysis

Q1.i $O(N^2)$ **ii** $O(N \log_2 N)$ **Q2.i** $O(2^N)$ **ii.** $O(N^2)$

Q3. i A **ii** C **iii.** A

inner loop iterates N times, outer loop iterates (take N/2 times N), so Big-O is $N * N = \frac{N^2}{2}$

$O(\frac{N^2}{2}) = O(N^2)$

outer loop iterates N times, inner loop iterates $\log_2 N$ times because j is divided by 2 for each iteration. Big-O is $N * \log_2 N$

Inner loop:

$2^{iterations - 1} = N$

$\log_2 N = iterations - 1$

$\log_2 N = iterations$

CMPSC 24 Spring 2018 Midterm-II Answer Sheet

Name: [redacted]____________ Seat no. _______ Perm no. :______________

Student to your left:_____________________ Student to your right _____________________

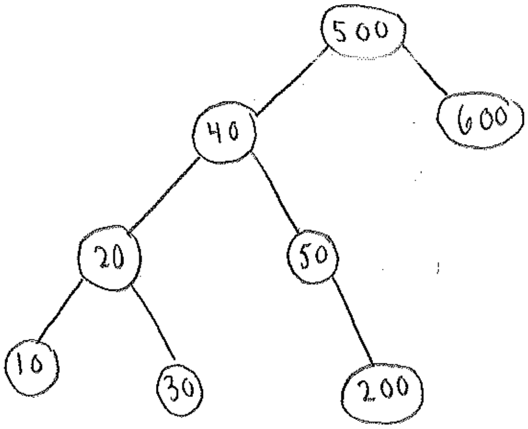
Part 3: Q1 [10 pts[

i. no **ii** yes **iii** A5 **iv** A10 **v.** A11 **vi.** A1

vii a A5 A3 A9 A4 A8 A1 A7 A2 A11 A6 A10

vii b A5 A9 A8 A4 A3 A7 A11 A10 A6 A2 A1

Q2 [10 pts] Insert 500, 40, 20, 30, 10, 50, 600, 200 into a BST that is initially empty BST in the provided order. Draw the final BST



CMPSC 24 Spring 2018 Midterm-II Answer Sheet

Name: [redacted]________________ Seat no. _______ Perm no. :________________

Student to your left:_______________________ Student to your right _______________________

Q3 [Extra credit: 10 pts]

```
int BST:: get Height () const {
      return get Height Helper (root);
 }


     BST::
int ^get Height Helper ( Node * n) {
    if (n== NULL)
        return -1;
    int left Height = getHeight Helper (n->left);
    int right Height = get Height Helper ( n-> right);
    if ( left Height > right Height)
        return 1+ left Height;
    else
        return 1 + right Height;

 }
```

CMPSC 24 Spring 2018 Midterm-II Answer Sheet

Name: [redacted]________________ Seat no. _______ Perm no. :_______________

Student to your left:______________________ Student to your right ______________________