1. (10pt) Do Big-Oh Analysis for the following functions:

1) $T(n)=100\log n+10n+2500$
2) $T(n)=(\log n)^{20}+1000n^{0.05}$

Solution: 1) O(n)
2) $O(n^{0.05})$

2. (12 pt) Analyze the running time of the following function with i = 1 and j = n. Write the recursive relation for T(n) and derive the Big-O for it.

```
void function(int[] a, i ,j) {
    int sum = 0;
    if (i == j) return;
    function(a, i, i + (j - i) / 2);
    function(a, i + (j - i) / 2, i + 3 * (j - i) / 4);
    function(a, i + 3 * (j - i) / 4, j);
    for (int k = i; k <= j; k++)
        sum = sum + a[k];
}
```

Solution: $T(n) = T(n/2) + 2T(n/4) + n$, $O(n\log n)$

3. (20 pt) Is the following statement true or false? (If false, give the correct statement)

1) In a binary search tree, the largest key can only be stored in a leaf node.
2) When using pair comparison to find the maximum and minimum value of n number, log(n) rounds are needed at most.

3) Suppose program 1 has best case running time $T_1(n)=3n^2$, program 2 has best case running time $T_2(n)= 10n$ and they can output the same results, we can choose the second for better performance.

4) In order to delete one node in a doubly linked list, we need to do four link changes in total.

Solution: 1) False, the node with one child node is also possible
2) True
3) False, we need worst running time to choose the better one
4) False, only two link changes are needed

4. (12pt) Rotate Singly-linked List.

Given a singly-linked list and an integer k. Define an operation named “Rotation”, which means moving the tail of the list to the head position. For example,
Linked List: [2] -> [4] -> [5] -> [1]. K=1
After “Rotation” for K times, we will have [1] -> [2] -> [4] -> [5].
If we change the K from 1 to 3, The linked list would become:

[4] -> [5] -> [1] -> [2]

The definition of List Node in C++ is as following:

```
struct ListNode {
    int val;
    ListNode *next;
    ListNode() : val(0), next(nullptr) {}
    ListNode(int x) : val(x), next(nullptr) {}
    ListNode(int x, ListNode *next) : val(x), next(next) {}
};
```

a. Please complete the following function named “rotate”.
It’s okay to write pseudo code, but the pointers operations need to be correct. (Pointer operation is the core of linked list.)
The return value and parameters are provided. (The function should return the head pointer after rotation.) **Boundary conditions need to be considered and shown in your code.**

Ans:

```
ListNode * rotate(ListNode* head, int k){

//Please write your code here.

}
```

b. What is the complexity in the last question? Is it the lower bound of this function? If not, what’s the lower bound?

Solution:

a.

```
ListNode * rotate(ListNode* head, int k){
        if (k == 0 || head == nullptr || head->next == nullptr) return head;
        int n = 1;
        ListNode* iter = head;
        while (iter->next != nullptr) {
                iter = iter->next;
                n++;
        }
        int add = n - k % n;
        if (add == n) return head;
        iter->next = head;
        while (add--) iter = iter->next;
        ListNode* ret = iter->next;
        iter->next = nullptr;
        return ret;
}
```

b. O(n). Yes.

5. (12pt)
a. Insert the following numbers into a hash table of size 7 using quadratic probing: 7, 15, 22, 34, 45, 51. The hash function is h(x)=x%7. After inserting all the numbers, what is the load factor of the hash table now?
b. Can we insert the above numbers into a hash table of size 6? If not, why?

Solution:
a.

0	1	2	3	4	5	6
7	15	22	45	51		34

The load factor is 6/7.
b. No, we cannot. Since the number 51 will always attempt to insert into the occupied slots.

6. (12pt) Monotonic stack is a type of stack that all elements in the stack are sorted. Consider a monotonic stack such that the value of the top stack element is always larger than other elements. If pushing an unordered element into this stack, the stack firstly pops the elements with a value larger than the unordered element, and then pushes the unordered element, and other elements that are popped before, into the stack again. This ensures the monotonicity of the stack.

a. Given a sequence of elements 70, 61, 20, 76, 97, 9. Please calculate the number of push and pop operations to this monotonic stack.

b. Now consider a sequence with N elements, what is the maximum number of push and pop operations respectively?

Solution: a. Push: 14; Pop: 8
b. Push: N(N+1)/2; Pop: N(N-1)/2

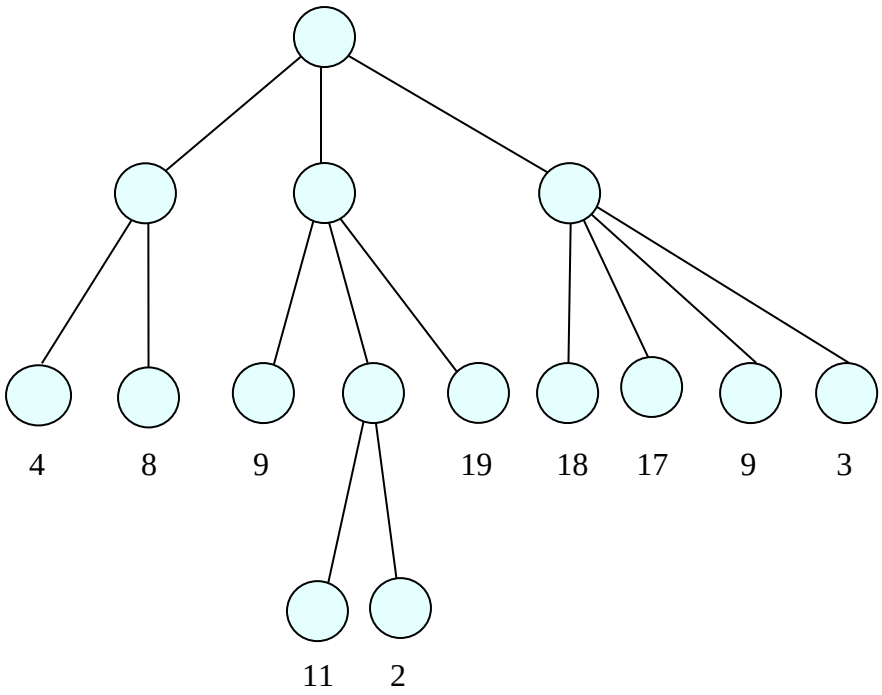
7. (10pt)

Given the following traversal sequences, show the preorder traversal of the corresponding binary tree:

Postorder sequence: ADBFCGE
Inorder sequence: AFDBEGC

Solution: Preorder sequence: EFABDGC

8. (12pt)
You are given the following game tree where player 1 takes the first move
and the total amount of 20 dollars will be divided between player 1 and player 2 after the game ends. The value specified for a certain leaf is the amount of dollars player 1 can get if the game process follows the path from root to that leaf. Please compute the final result for the player 1 and player 2. Does this tree allow you to cut some branches using α - β pruning if you search the children from left to right? List the cut branches if exist.



Solution: Player 1 get 9 and player 2 get 11. Alpha - beta cut: 3, 2.