Question 1

Consider the activity selection problem discussed in the lectures. For each greedy choice below, decide if it would yield the optimum solution by either giving a proof, or a counterexample.

a) Selecting the activity with least duration.

b) Selecting the activity that overlaps the fewest remaining activities.

c) Selecting the activity with the earliest start time.

d) Selecting the activity with the latest start time.

Question 2

A palindrome is a sequence of characters that reads the same backward as forward. In other words, if you reverse the sequence, you get the same sequence. For example, "a", "bb", "aba", "ababa", "aabbaa" are palindromes. We also have real words which are palindromes, like "noon", "level", "rotator", etc.

We also have considered the concept of a subsequence in our lectures. Given a word A, B is a subsequence of A, if B is obtained from A by deleting some symbols in A. For example, the following are some of the subsequences of the sequence "*abbdcacdb*": "*da*", "*bcd*", "*abba*", "*abcd*", "*bcacb*", "*bbccdb*", etc.

The Longest Palindromic Subsequence (LPS) problem is finding the longest subsequences of a string that is also a palindrome. For example, for the sequence "*abbdcacdb*", the longest palindromic subsequence is "*bdcacdb*", which has length 7. There is no subsequence of "*abbdcacdb*" of length 8 which is a palindrome.

One can find the length of LPS of a given sequence by using dynamic programming. As common in dynamic programming, the solution is based on a recurrence.

Given a sequence $A = a_1 a_2 \dots a_n$, let $A[i, j]$ denote the sequence $a_i a_{i+1} \dots a_j$. Hence it is part of the sequence that starts with a_i and ends with a_j (including these symbols). For example, if $A = abcdef$, $A[2, 4] = bcd$, $A[1, 5] = abcde$, $A[3, 4] = cd$, etc.

For a sequence $A = a_1 a_2 \dots a_n$, let us use the function $L[i, j]$ to denote the length of the longest palindromic subsequence in $A[i, j]$.

(a) If we have a sequence $A = a_1 a_2 \dots a_n$, for which values of i and j, $L[i, j]$ would refer to the length of the longest palindromic subsequence in A?

(b) Write the recurrence for $L[i, j]$.

$$L[i,j] = \begin{cases} 0 & \text{if } i > j \\ \dots\dots\dots\dots\dots\dots\dots & \text{if } i = j \\ \dots\dots\dots\dots\dots\dots\dots & \text{if } i < j \text{ and } a_i = a_j \\ \dots\dots\dots\dots\dots\dots\dots & \text{if } i < j \text{ and } a_i \neq a_j \end{cases}$$

(c) What would be the worst case time complexity of the algorithm in Θ notation? Why?

(d) Considering a memoization-based approach, complete the following pseudocode for the dynamic programming solution to the LPS problem. Assume the existence of a 2D array 'dp' of size $n \times n$ initialized to -1, where n is the length of sequence A, and a function 'LPS(i, j)' that computes the length of the longest palindromic subsequence in $A[i, j]$.

```
function LPS(i, j):
    if dp[i][j] != -1:
        return ...
    if i > j:
        dp[i][j] = 0
    elif i == j:
        dp[i][j] = ...
    elif A[i] == A[j]:
        dp[i][j] = ...
    else:
        dp[i][j] = max(LPS(..., j), LPS(i, ...))
    return dp[i][j]
```

Question 3

To compute the depth of a given node in an RBT in constant time, consider augmenting the depths of nodes as additional attributes in the nodes of the RBT. Please explain by an example why this augmentation increases the asymptotic time complexity of inserting a node into an RBT in the worst case.

Question 4

Write pseudocode for LEFT-ROTATE that operates on nodes in an interval tree and updates the max attributes in $O(1)$ time.