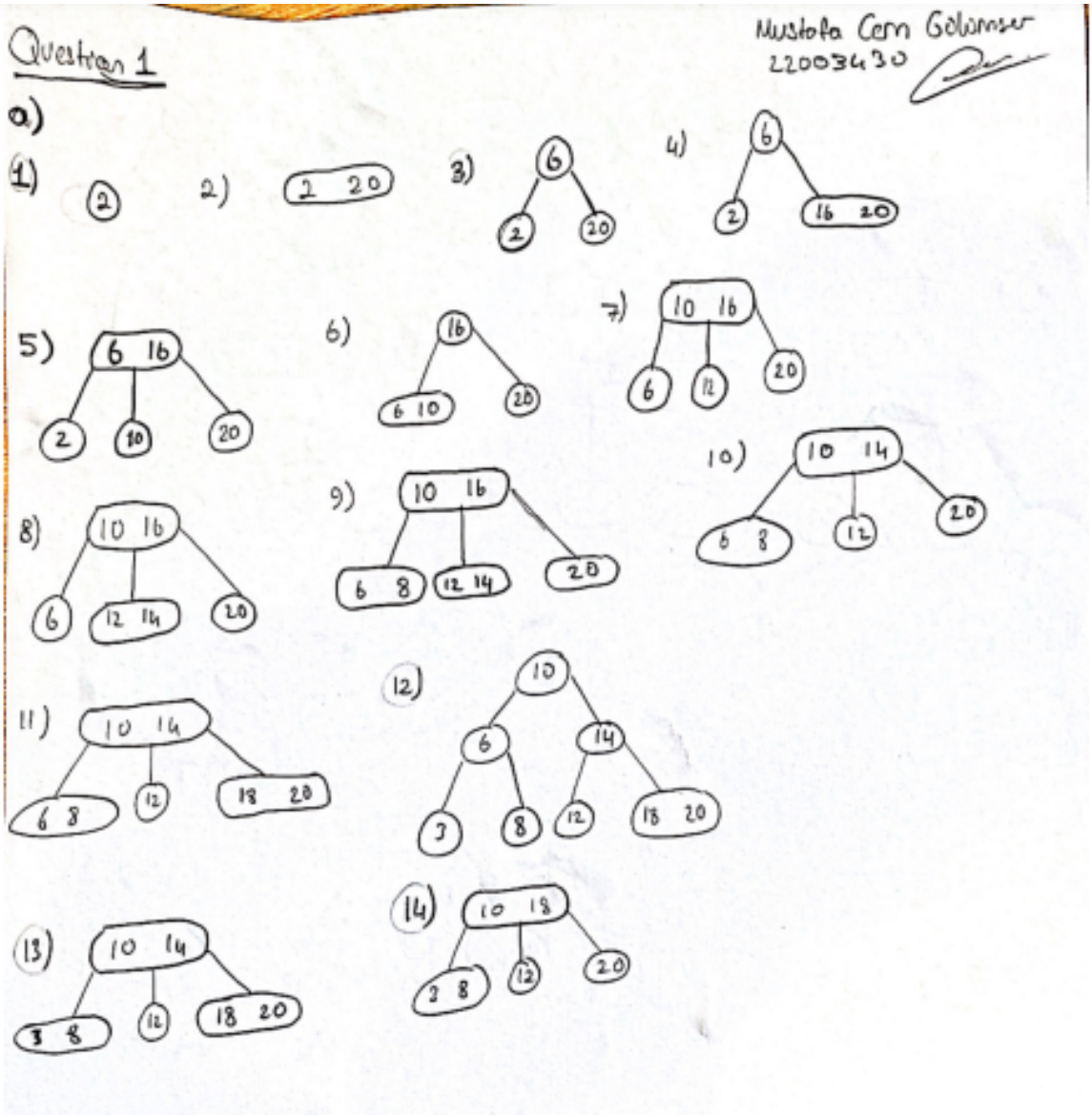
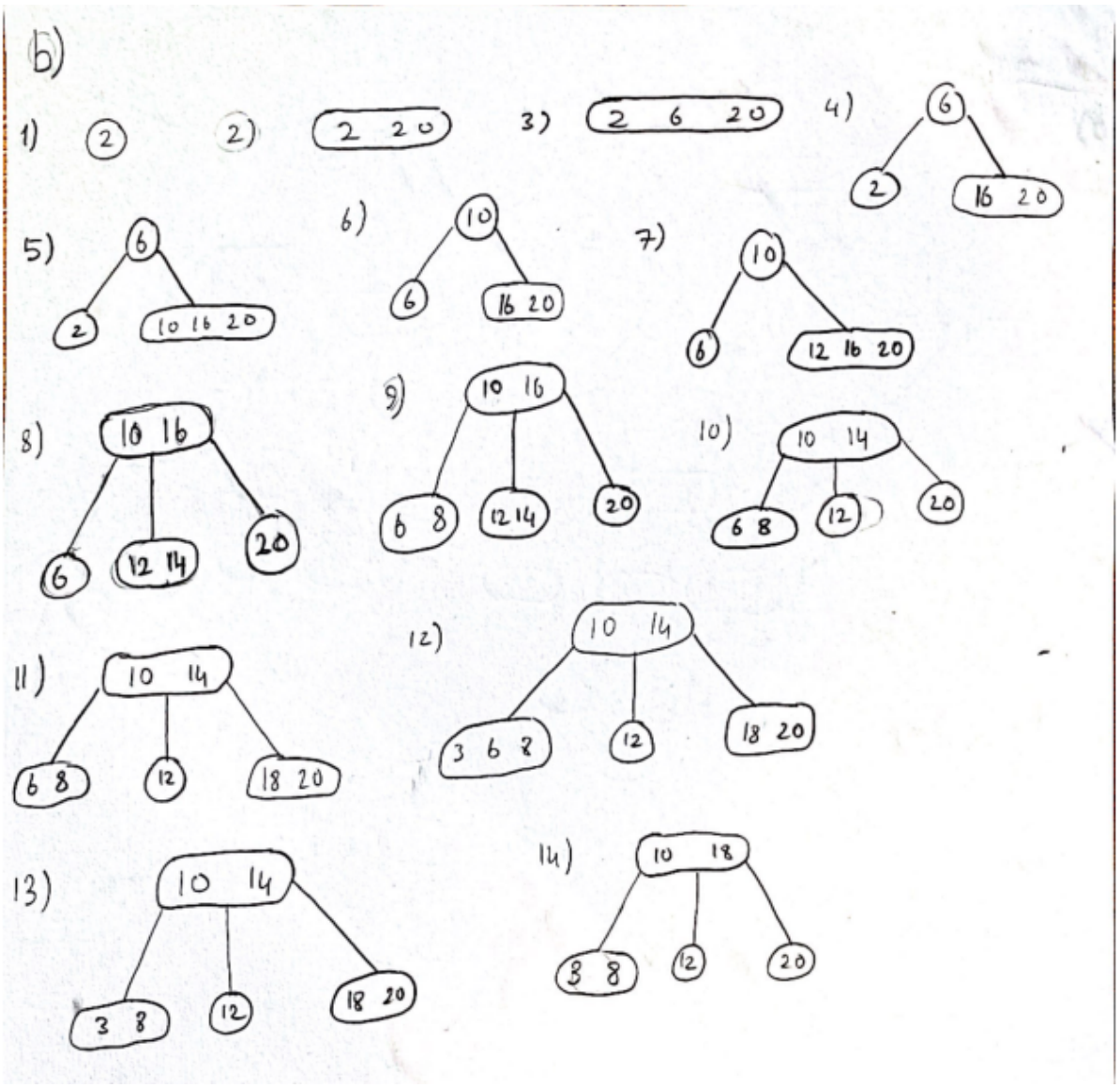
CS-202 - 02
Homework 4
Mustafa Cem Gülümser
22003430
05.05.2022

Question 1:

a)

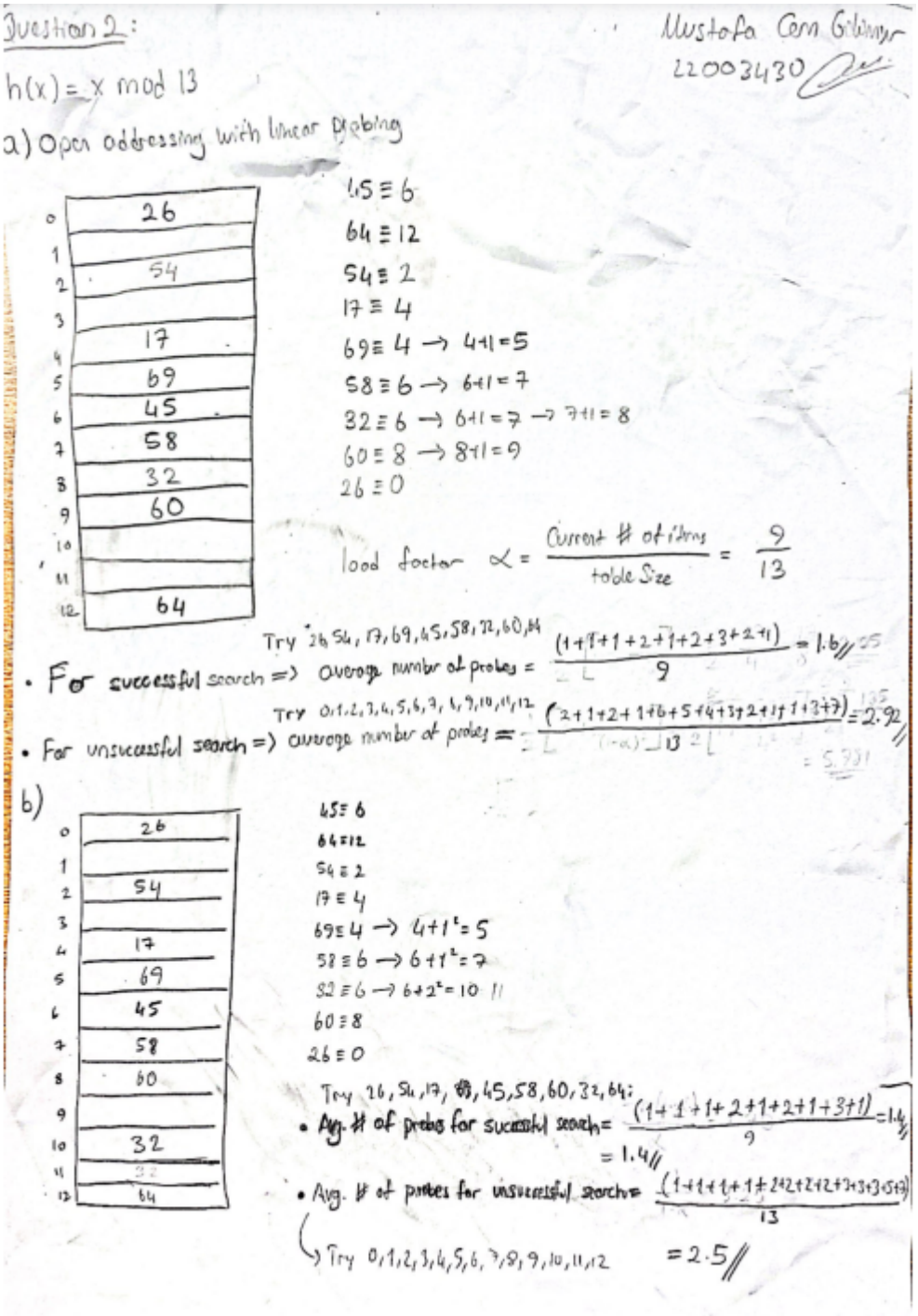


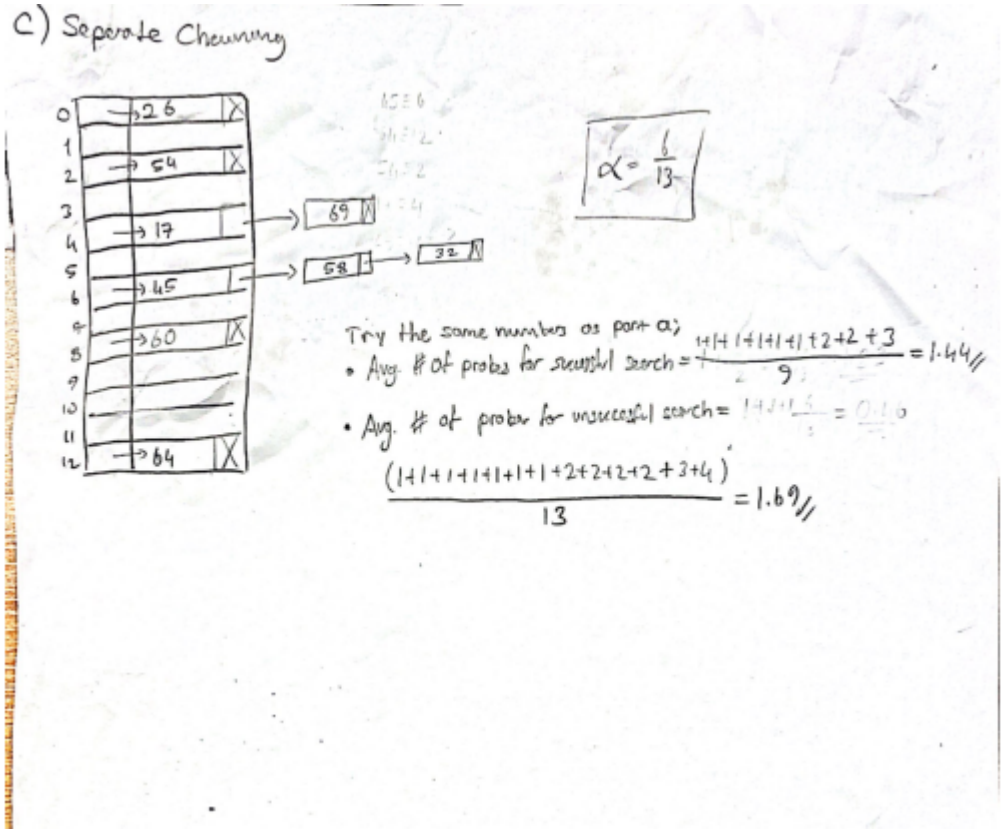
b)


b)
1) 2
2) 2 20
3) 2 6 20
4) 6 2 16 20
5) 6 2 10 16 20
6) 10 6 16 20
7) 10 6 12 16 20
8) 10 16 6 12 14 20
9) 10 16 6 8 12 14 20
10) 10 14 6 8 12 20
11) 10 14 6 8 12 18 20
12) 10 14 3 6 8 12 18 20
13) 10 14 3 8 12 18 20
14) 10 18 3 8 12 20

Question 2:

a,b,c) Please do not mind the load factor calculations.





Question 3:
b)

Asymptotic worst-case behaviors for the following functions:

- Insertion:

Considering that the implementation of the graph is an adjacency list, for each vertex, we have its adjacent airports stored in a linked list. To locate the vertex we will make insertion to, the complexity is always O(1) because the linked list pointers are stored in an array. Let us assume that we have N airports in the graph, and that the airport we want to make an insertion to is adjacent to all other N-1 airports. In this worst case, the algorithm will trace the list until its end to report that the flight already exists. Therefore, the worst case complexity is O(1) + O(N-1) = O(N).

- List:

The list works in a similar way with the insertion algorithm. It locates the vertex to be listed and traverses the entire linked list to display the existing flights. Again in the worst case, the selected airport will be adjacent to all other N-1 airports. Hence the complexity of the list algorithm at worst case becomes O(N).

- Shortest Path:

To find the shortest path between two arbitrary airports (let us call them airport1 and airport2) in a graph with N airports, we utilize Dijkstra's shortest path algorithm. However, in addition to this, we also include a path array to keep the path from airport1 to airport2. In the first step of the algorithm, we initialize the weight array, and the complexity of this loop is independent of the worst case, which is O(N). The following loop is executed N-1 times and contains inner loops. Within this loop, we first execute a loop to find the smallest weight edge from airport1 to an adjacent vertex v such that v is not in the vertex set. At the worst case, each vertex will be adjacent to all other N-1 vertices (in other words, the graph will be complete) where this loop will be executed N times. The next loop is also executed N times, which updates the weight and path array if a shorter path is found. Hence, the complexity becomes: $O(N) + O(2(N-1)*N) = O(N^2)$.

- Minimize Cost

To minimize the cost of flights, we find the minimum spanning tree using Prim's algorithm. To print the initial and final cost, this algorithm calls another function named findTotalCost, which runs at $O(N^2)$ time. After printing, this function initializes another adjacency list to be replaced with the older list after the minimization, this initialization loop runs for N times. The main body of the function contains a while loop which runs until there are no more unvisited edges, which is independent of the worst case and runs for N-1 times. Within the while loop, there is a double for loop to find the least cost edge in the entire graph. In the worst case where the graph is complete, the loop executes for each edge, which is $N*(N-1)/2$ times. Hence, the overall complexity of the algorithm at worst case becomes: $O(2N^2) + O(N) + O(N*N*(N-1)/2) = O(N^3)$.