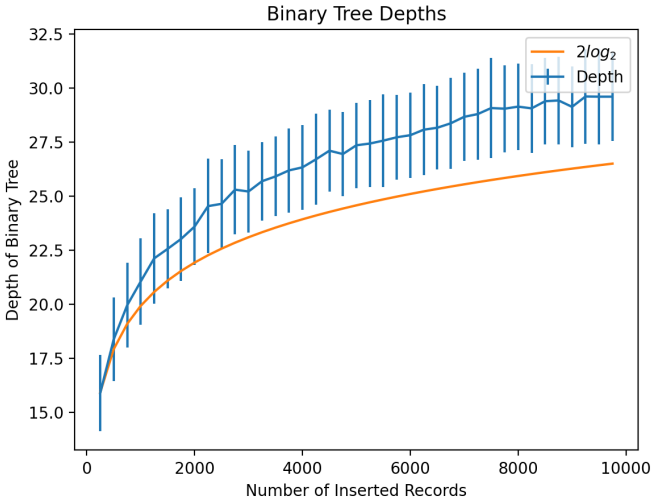
Programming Assignment 4

You are allowed to work in groups. Give all group members and certify that all group members worked on all problems.

Problem 1:



Install (with `pip3.13 install numpy` and `pip3.13 install matplotlib`) the Python modules NumPy and Matplotlib.

Using the presentation, implement binary search trees. We then use this implementation in order to get the depth in dependence on the number of records inserted into a binary search tree. You can use the code below, which will take quite some time to run. You might want to replace 200 with 10 for debugging purposes.

```
if __name__ == '__main__':
    pts = list(range(250, 10000, 250))
    means, stds = [], []
    for l in pts:
        results = []
        for j in range(200):
            b = Binary_Tree()
            for i in range(l):
                x = random.randint(0, 2**16-1)
                b.insert(x)
            results.append(b.depth())
        means.append(np.mean(results))
        stds.append(np.std(results))

    my_plot = plt.errorbar(pts, means, stds, label='Depth')
    plt.plot(pts, [2*math.log(p,2) for p in pts], label='$2 log_2$')
    plt.title('Binary Tree Depths')
    plt.xlabel('Number of Inserted Records')
```

```
    plt.ylabel('Depth of Binary Tree')
    plt.legend()

    plt.show()
```

Problem 2:

Implement a linear hash table with linear probing. Your insert function should return the number of slots visited (or -1 in case the insert fails). The keys are integers and the hash of an integer is just the remainder of the integer divided by the number of slots.

Then create a hash table with 1024 slots. Insert as many records with random keys as to obtain a certain occupancy level. Then insert another 10 records counting the number of slots visited per insert and print them out.

Choose occupancies of (rounded down) of 1/2, 2/3, 3/4, 4/5, 5/6, 6/7, 7/8, 8/9, 9/10 and display the results. For example, one of my runs gave

```
0.5 [2, 2, 1, 1, 2, 1, 1, 2, 5, 2]
0.666015625 [1, 7, 4, 28, 3, 1, 5, 1, 4, 1]
0.75 [20, 6, 2, 10, 10, 3, 10, 2, 3, 9]
0.7998046875 [8, 10, 10, 9, 11, 20, 8, 22, 7, 11]
0.8330078125 [3, 55, 5, 18, 12, 3, 13, 1, 4, 1]
0.8564453125 [19, 3, 7, 58, 16, 74, 6, 7, 11, 1]
0.875 [6, 11, 1, 33, 50, 2, 3, 4, 1, 1]
0.888671875 [34, 15, 17, 41, 19, 1, 6, 85, 21, 1]
0.8994140625 [45, 44, 17, 8, 8, 3, 4, 3, 88, 49]
```

As you can see, inserts become increasingly cost prohibitive as the occupancy increases.