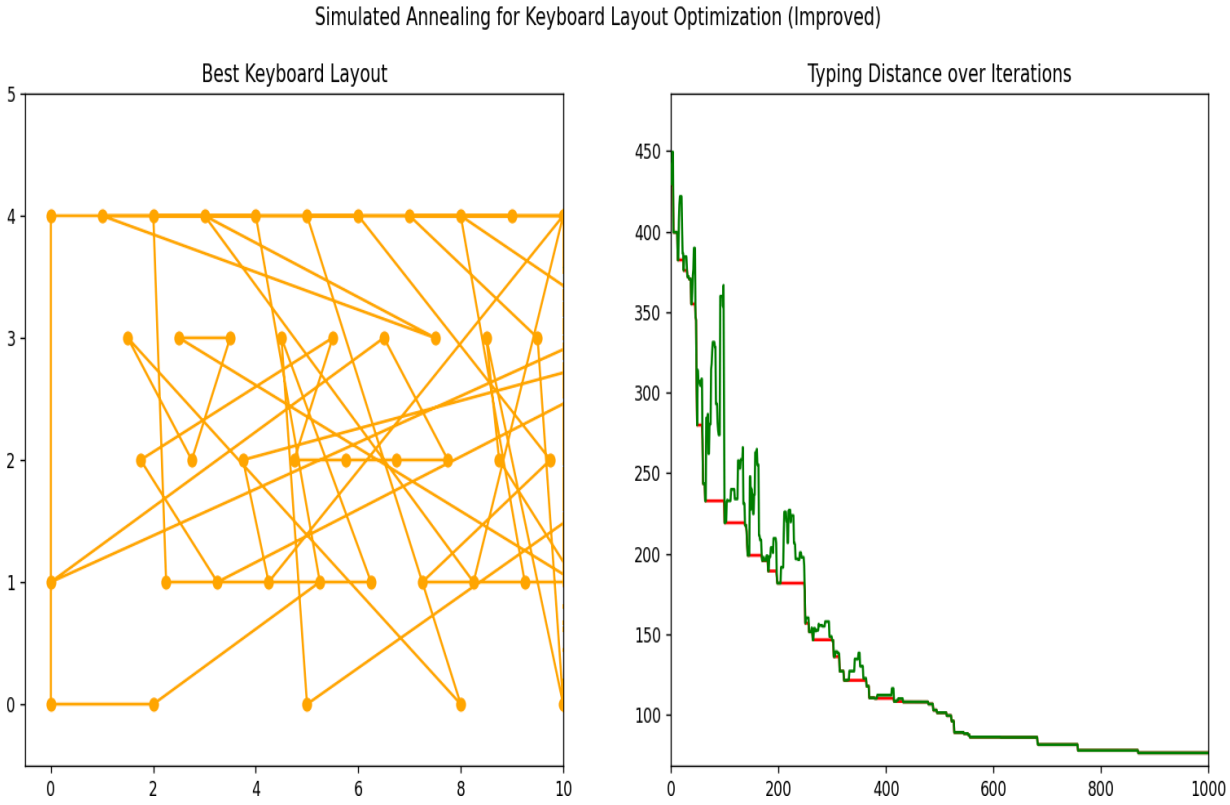
# ASSIGNMENT 5



SHASHWAT GAUTAM

EE23B150

# MAIN ASSUMPTION AND ADDED FUNCTIONALITY:

- My assignment is based the qwerty_layout given by Sir in the Programming Quiz for keyboard
- I have used the start key to travel to the specific key while calculating the Euclidean Distance  using its formula
- I have added the functionality:
    i. Animation to the plot
    ii. 2 spacebar for ergonomic concern

# LOGIC SUMMARY

- ## KEYBOARD LAYOUT:

  I have used the QWERTY keyboard provided by sir  as my default keyboard.**key_positions Dictionary**, containing the positions of each key on the keyboard in (x, y) coordinates. Each key is mapped to,'pos' indicating the location of the key on a virtual 2D grid while "Start' is used to define the home row keys from which the distance calculation for that key is started.**characters Dictionary** is used to map each character (uppercase, lowercase, and symbols) to its required keys.

- ## GENERATE RANDOM LAYOUT:

  The function takes a list of keys as input , creates a copy of keys and  generates a random starting layout by shuffling the keys. This serves as the initial configuration for the simulated annealing process.

- ## CALCULATING EUCLIDEAN DISTANCE BETWEEN THE KEYS:

  Firstly I have stored the keys of random layout in key_position which basically gives a unique address to each key of randomized layout and set total_distance to 0 this basically keeps track of distance , the loop iterates over `text`, considering each consecutive

character pair (`char_1`, `char_2`) , if `char_1` and `char_2` are in `characters`, then retrieves the key sequences (`keys_1`, `keys_2`) for each then it loops over each possible key required to type each character, calculating distances between `k1` and `k2` in `layout_map` by Identifying the starting keys (`start_key_1`, `start_key_2`) for each character and their positions. Calculates distances (`dist_1` and `dist_2`) from the start positions to the actual key positions using Euclidean distance.

- **FINGER POSITION:**

  Here the the start key given in the layout simplifies the key position scenario , heifer we know that fingers are placed on home rows also the keys used as start key are mainly home row this end the need to define the keys occupancy finger wise instead we use start directly

- **GENERATE NEIGHBOR:**

  get_neighbour function creates a new layout by swapping 2 or 3 keys from current_layout ,solutions that are close to the existing solution, but different in some way, in order to evaluate whether the new solution is better than the existing one.

- **SIMULATED ANNEALING:**

  simulated_annealing initializes the optimization by generating a random current_layout and making a copy as best_layout.At each iteration, a new neighbor is generated. The cost of this is calculated, and the decision to *accept* or *reject* the new solution is based on either actual improvement, or a *coin toss* - a random event whose probability decreases with iterations. Eventually, the chance of accepting poorer solutions is so low that we settle to a solution, which hopefully is a good solution given the method used for searching through the neighborhood.

- **UPDATE KEYBOARD LOGIC:**

  The update_keyboard function updates the keyboard layout and distances in each animation frame the orange color animation . It gets the coordinates for each key and updates plot data.

- **STORAGE OF OPTIMIZED KEYS:**

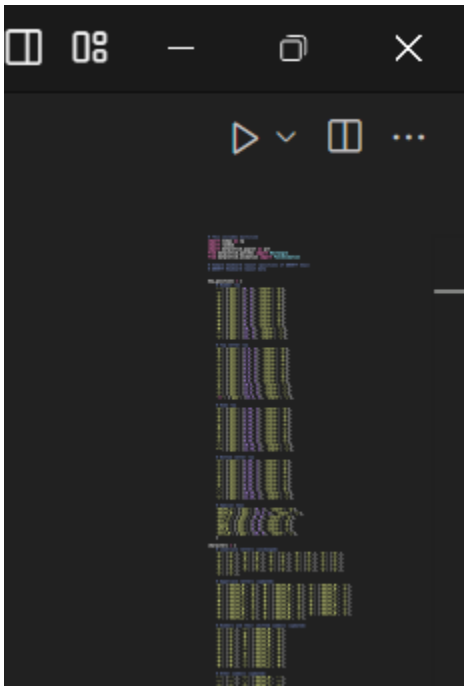
  This function stores the keys of optimized keyboard these optimized keys are used to generate the final optimized keyboard plot

- **DEFINING MAIN FUNCTION:**

The main function is used for running the functions I made so far. It first collects user input, initializes parameters like temperature and iterations, and calls the simulated_annealing function to perform the optimization which returns the best layouts and corresponding typing distances at each step. Then using  matplotlib, it plots two subplots: one for the evolving keyboard layout and one for tracking typing distance over time. Using an animated plot, it updates the layout and typing distance in real-time, showing the progress of optimization. In the end, the function displays an animated figure where the keyboard layout and typing distance adjust iteratively.

# RUNNING THE CODE:

1. Unzip the file it contains three files namely:
    a. README.pdf
    b. EE23B150_A5.py
    c. custom_layout.py
2. Open the VS Code and upload EE23B150_A5.py
3. Just run by click on this play  button



4. Or you can open the terminal and type “python EE23B150_A5.py”
5. Press “Enter”
6. {The code works perfectly on VS code but when running on Jupyter NB the animation control appear for keyboard layout instead of plot of current distance}