Practical 6 - Write a program to implement the Activity Selection Problem using the Greedy Method. Given n activities with their start and finish times, select the maximum number of non-overlapping activities that can be performed by a single person, assuming a person can only work on one activity at a time.

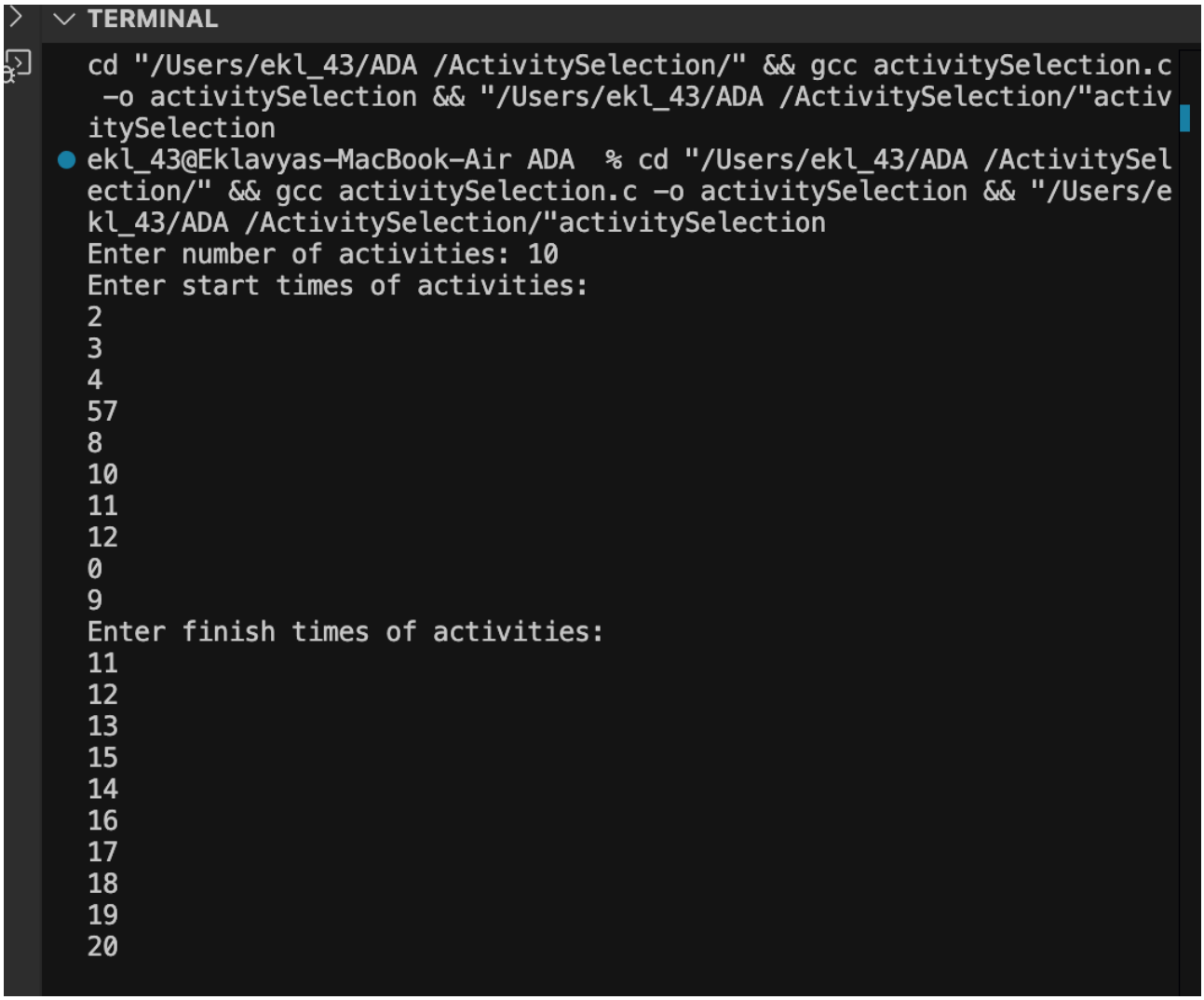
Activity Selection Problem

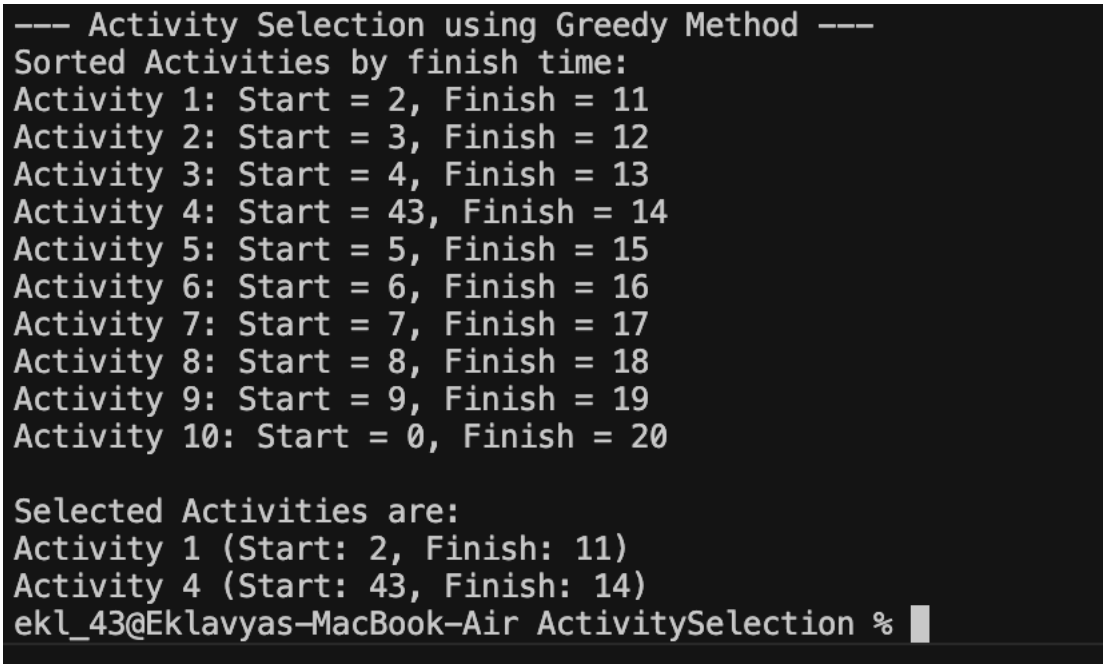
C code:

```
#include <stdio.h>

// Function to swap two integers
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

// Function to sort activities by finish time using simple Bubble Sort
void sortActivities(int start[], int finish[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (finish[j] > finish[j + 1]) {
                swap(&finish[j], &finish[j + 1]);
                swap(&start[j], &start[j + 1]);
            }
        }
    }
}

// Function to perform Activity Selection
void activitySelection(int start[], int finish[], int n) {
    // Sort activities by finish time
    sortActivities(start, finish, n);

    printf("Sorted Activities by finish time:\n");
    for (int i = 0; i < n; i++) {
        printf("Activity %d: Start = %d, Finish = %d\n", i + 1, start[i], finish[i]);
    }

    printf("\nSelected Activities are:\n");

    int i = 0;
    printf("Activity %d (Start: %d, Finish: %d)\n", i + 1, start[i], finish[i]);

    // Consider rest of the activities
    for (int j = 1; j < n; j++) {
        if (start[j] >= finish[i]) {
            printf("Activity %d (Start: %d, Finish: %d)\n", j + 1, start[j], finish[j]);
            i = j;
        }
    }
}

int main() {
```

```
    int n;

    printf("Enter number of activities: ");
    scanf("%d", &n);

    int start[n], finish[n];

    printf("Enter start times of activities:\n");
    for (int i = 0; i < n; i++)
        scanf("%d", &start[i]);

    printf("Enter finish times of activities:\n");
    for (int i = 0; i < n; i++)
        scanf("%d", &finish[i]);

    printf("\n--- Activity Selection using Greedy Method ---\n");
    activitySelection(start, finish, n);

    return 0;
}
```

Output:





Conclusion:

In this practical, the **Activity Selection Problem** was successfully implemented using the **Greedy Method** in C. The program selects the maximum number of non-overlapping activities that can be performed by a single person, given the start and finish times of all activities.

The greedy approach works efficiently by first **sorting the activities based on their finish times** and then **selecting activities that start after or when the previous one finishes**. This ensures the optimal solution with minimal computation.

Through this practical, we learned how **greedy algorithms** make locally optimal choices at each step to reach a **globally optimal solution**, demonstrating an important concept in algorithm design and optimization.