PRACTICAL NO: 01

Name : Soham Adgokar

Roll No : A2-B2-28

Date : 22/07/25 (5.00 PM)

Git-hub link : https://github.com/sohamadgokar

TASK A)

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
float generateRdata(float min, float max) {
    return min + ((float) rand() / RAND_MAX) * (max - min);
}
void generateData(float temp[], float pressure[], int n) {
    for (int i = 0; i < n; i++) {
        temp[i] = generateRdata(-20, 50);
        pressure[i] = generateRdata(950, 1050);
    }
```

```
}
int findMinTemperature(float temp[], int n) {
    int minIndex = 0;
    for (int i = 1; i < n; i++) {
        if (temp[i] < temp[minIndex]) {
            minIndex = i;
        }
    }
    return minIndex;
}
int findMaxPressure(float pressure[], int n) {
    int maxIndex = 0;
    for (int i = 1; i < n; i++) {
        if (pressure[i] > pressure[maxIndex]) {
            maxIndex = i;
        }
    }
    return maxIndex;
}
int main() {
    int n = 100;
    float temp[n], pressure[n];
```

```
    generateData(temp, pressure, n);
    clock_t start, end;
    double duration;
    start = clock();
    int minTempIndex = findMinTemperature(temp, n);
    end = clock();
    duration = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Minimum Temperature: %.2f °C , Time: %lf
seconds\n", temp[minTempIndex], duration);
    start = clock();
    int maxPressureIndex = findMaxPressure(pressure, n);
    end = clock();
    duration = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Maximum Pressure: %.2f hPa ,Time: %lf seconds\n",
pressure[maxPressureIndex], duration);


    return 0;
}
```

Task B)

Code:

```
#include <stdio.h>
```

```
#include <stdlib.h>
#include <time.h>
float generateRdata (float min, float max) {
    return min + ((float) rand() / RAND_MAX) * (max - min);
}

void generateData(float temp[], float pressure[], int n) {
    for (int i = 0; i < n; i++) {
        temp[i] = generateRdata (-20, 50);
        pressure[i] = generateRdata (950, 1050);
    }
}
int NFindMin(float arr[], int n) {
    for (int i = 0; i < n; i++) {
        int isMin = 1;
        for (int j = 0; j < n; j++) {
            if (arr[j] < arr[i]) {
                isMin = 0;
                break;
            }
        }
        if (isMin)
```

```
            return i;
        }
        return -1;
    }
    int NFindMax(float arr[], int n) {
        for (int i = 0; i < n; i++) {
            int isMax = 1;
            for (int j = 0; j < n; j++) {
                if (arr[j] > arr[i]) {
                    isMax = 0;
                    break;
                }
            }
            if (isMax)
                return i;
        }
        return -1;
    }
    int main() {
        int n = 100;
        float temp[n], pressure[n];
        generateData(temp, pressure, n);
```

```
    clock_t start, end;
    double duration;
    start = clock();
    int minTempIndex = NFindMin(temp, n);
    end = clock();
    duration = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Minimum Temperature: %.2f °C ,Time: %lf
seconds\n", temp[minTempIndex], duration);
    start = clock();
    int maxPressureIndex = NFindMax(pressure, n);
    end = clock();
    duration = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Maximum Pressure: %.2f hPa ,Time: %lf seconds\n",
pressure[maxPressureIndex], duration);
    return 0;
}
```

Output :

TASK	Loop Type	Time complexity	Parameters	n=100	n=10000	n=1000000
TASK-A	LINEAR	O(n)	Temperature Pressure	Time: 0.000002 seconds Time: 0.000001 seconds	Time: 0.000004 se Time: 0.000004 se	Time: 0.001506 sec Time: 0.001472 sec
TASK-B	Quadratic	O(n^2)	Temperature Pressure	Time: 0.000004 s Time: 0.000004 s	Time: 0.000037 se Time: 0.000055 se	Time: 0.010079 se Time: 0.038329 se

TASK C)

Code (Linear Search) :

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void generateSortedData(float arr[], int n) {
    float first = 30.0 / n;
    for (int i = 0; i < n; i++) {
        arr[i] = 20.0 + i * first;
    }
}
int linearSearch(float arr[], int n) {
    for (int i = 0; i < n; i++) {
        if (arr[i] >= 30.0)
            return i;
    }
    return -1;
}
int main() {
    int n = 100;
```

```
    float temp[n];
    generateSortedData(temp, n);
    clock_t start = clock();
    int index = linearSearch(temp, n);
    clock_t end = clock();
    double timeTaken = (double)(end - start) /
CLOCKS_PER_SEC;
    printf("First temperature >= 30°C at index %d: %.2f°C\n",
index, temp[index]);
    printf("Time taken: %lf seconds\n", timeTaken);


    return 0;
}
```

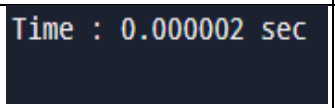
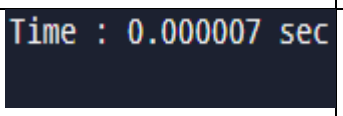
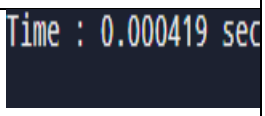
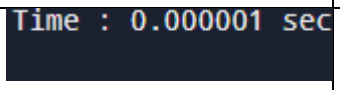
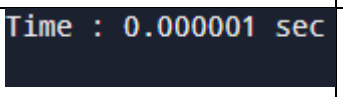
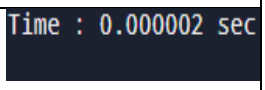
Code (Binary Search):

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void generateSortedData(float arr[], int n) {
    float first = 30.0 / n;
    for (int i = 0; i < n; i++) {
        arr[i] = 20.0 + i * first;
    }
```

```
}
int binarySearch(float arr[], int n) {
    int left = 0, right = n - 1, result = -1;

    while (left <= right) {
        int mid = (left + right) / 2;
        if (arr[mid] >= 30.0) {
            result = mid;
            right = mid - 1;
        } else {
            left = mid + 1;
        }
    }

    return result;
}

int main() {
    int n = 1000;
    float temp[n];
    generateSortedData(temp, n);
    clock_t start = clock();
```

```
    int index = binarySearch(temp, n);
    clock_t end = clock();
    double timeTaken = (double)(end - start) /
CLOCKS_PER_SEC;
    printf("First temperature >= 30°C at index %d: %.2f°C\n",
index, temp[index]);
    printf("Time : %lf seconds\n", timeTaken);


    return 0;
}
```

Output :

Algorithm	Time complexity	N=100	N=10000	N=1000000
Linear Search	O(n)			
Binary Search	O(log n)			

Conclusion:

1) Task A – Linear Search Approach

- A single-pass linear search is used to find the minimum temperature and maximum pressure.
- Time complexity is O(n), efficient for larger datasets.

2) Task B – Naive Approach

- Uses a naive double-loop method to compare every element with all others.
- Time complexity is $O(n^2)$, making it significantly slower for large datasets.

3) Task C – Linear vs Binary Search

- Applies linear search (O(n)) and binary search (O(log n)) to find the first temperature ≥ 30°C.
- Binary search requires the array to be pre-sorted but is much faster.