گزارش کار شماره ۴ آزمایش سیستم عامل

نام و نام خانوادگی:

محمد مهدی حائری 40131008

محمد علی آقایی 40131059

توزیع نرمال بدون استفاده از برنامه نویسی چند فرایندی و به صورت سری:

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

int main() {
    srand(time(NULL));
    int sampleCount = 500;
    int hist[25] = {0};

    clock_t start = clock();

    for (int i = 0; i < sampleCount; i++) {
        int count = 0;
        for (int j = 0; j < 12; j++) {
            int randomNumber = rand() % 100;
            if (randomNumber < 50) {
                count++;
            } else {
                count--;
            }
        }

        int index = count + 12;
        if (index >= 0 && index < 25) {
            hist[index]++;
        }
    }

    clock_t end = clock();
    double time_spent = (double)(end - start) / CLOCKS_PER_SEC;

    printf("Histogram:\n");
    for (int i = 0; i < 25; i++) {
        printf("%2d: ", i);
```

```
        for (int j = 0; j < hist[i]; j++) {
            printf("*");
        }
        printf("\n");
    }

    printf("Time taken: %f seconds\n", time_spent);
    return 0;
}
```

توزیع نرمال با استفاده از از برنامه نویسی چند فرایندی (برای macos):

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/mman.h>

#define NUM_PROCESSES 4 // Number of child processes
#define SAMPLE_COUNT 500
#define HISTOGRAM_SIZE 25

// Structure to hold the shared histogram
typedef struct {
    int hist[HISTOGRAM_SIZE];
} SharedHistogram;

// Function to perform the simulation for a portion of the samples
void simulate(int start_index, int end_index, SharedHistogram
*shared_hist) {
    srand(time(NULL) ^ getpid()); // Seed with different values for each
process

    for (int i = start_index; i < end_index; i++) {
        int count = 0;
        for (int j = 0; j < 12; j++) {
            int randomNumber = rand() % 100;
            if (randomNumber < 50) {
                count++;
            } else {
```

```
                count--;
            }
        }

        int index = count + 12;
        if (index >= 0 && index < HISTOGRAM_SIZE) {
            // Atomically increment the histogram count
            __sync_fetch_and_add(&shared_hist->hist[index], 1);
        }
    }
}

int main() {
    // Allocate shared memory for the histogram
    SharedHistogram *shared_hist = mmap(NULL, sizeof(SharedHistogram),
                                        PROT_READ | PROT_WRITE,
                                        MAP_SHARED | MAP_ANONYMOUS, -1, 0);
    if (shared_hist == MAP_FAILED) {
        perror("mmap failed");
        exit(EXIT_FAILURE);
    }

    // Initialize the shared histogram to zero
    for (int i = 0; i < HISTOGRAM_SIZE; i++) {
        shared_hist->hist[i] = 0;
    }

    clock_t start = clock();

    // Calculate the number of samples per process
    int samples_per_process = SAMPLE_COUNT / NUM_PROCESSES;
    int remainder = SAMPLE_COUNT % NUM_PROCESSES;

    // Create child processes
    pid_t pids[NUM_PROCESSES];
    int current_start = 0;
    for (int i = 0; i < NUM_PROCESSES; i++) {
        int current_end = current_start + samples_per_process + (i <
remainder ? 1 : 0);
        pids[i] = fork();
        if (pids[i] == -1) {
            perror("fork failed");
            exit(EXIT_FAILURE);
        } else if (pids[i] == 0) {
            // Child process
            simulate(current_start, current_end, shared_hist);
```

```
            exit(EXIT_SUCCESS);
        }
        current_start = current_end;
    }

    // Wait for all child processes to complete
    for (int i = 0; i < NUM_PROCESSES; i++) {
        waitpid(pids[i], NULL, 0);
    }

    clock_t end = clock();
    double time_spent = (double)(end - start) / CLOCKS_PER_SEC;

    printf("Histogram (Multiprocessing):\n");
    for (int i = 0; i < HISTOGRAM_SIZE; i++) {
        printf("%2d: ", i);
        for (int j = 0; j < shared_hist->hist[i]; j++) {
            printf("*");
        }
        printf("\n");
    }

    printf("Time taken (Multiprocessing): %f seconds\n", time_spent);

    // Unmap the shared memory
    if (munmap(shared_hist, sizeof(SharedHistogram)) == -1) {
        perror("munmap failed");
        exit(EXIT_FAILURE);
    }

    return 0;
}
```

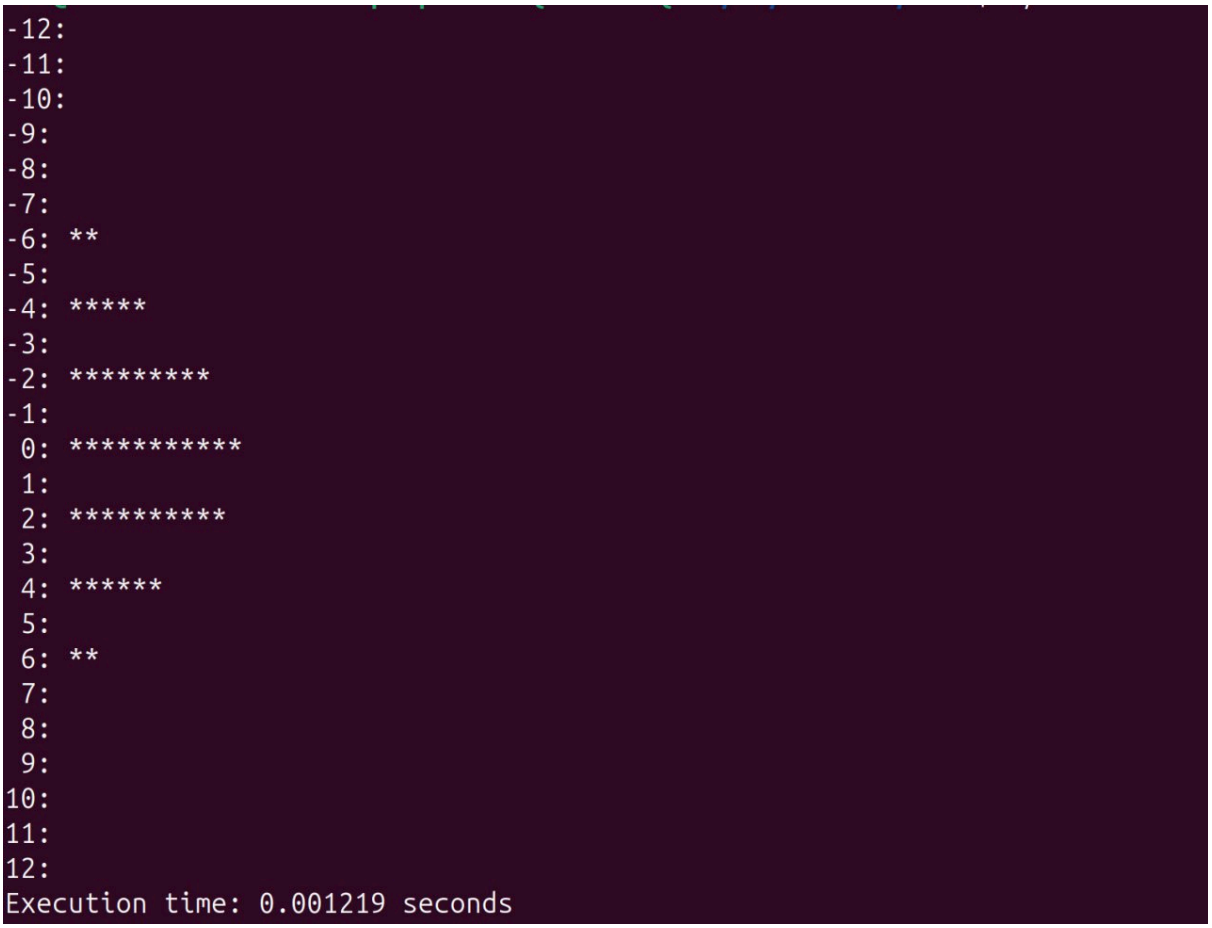
به صورت سری:

تعداد نمونه	۵۰۰۰	۵۰۰۰۰	۵۰۰۰۰۰
زمان اجرا	0.001219 seconds	0.012186 seconds	0.083818

به صورت چند فرایندی:

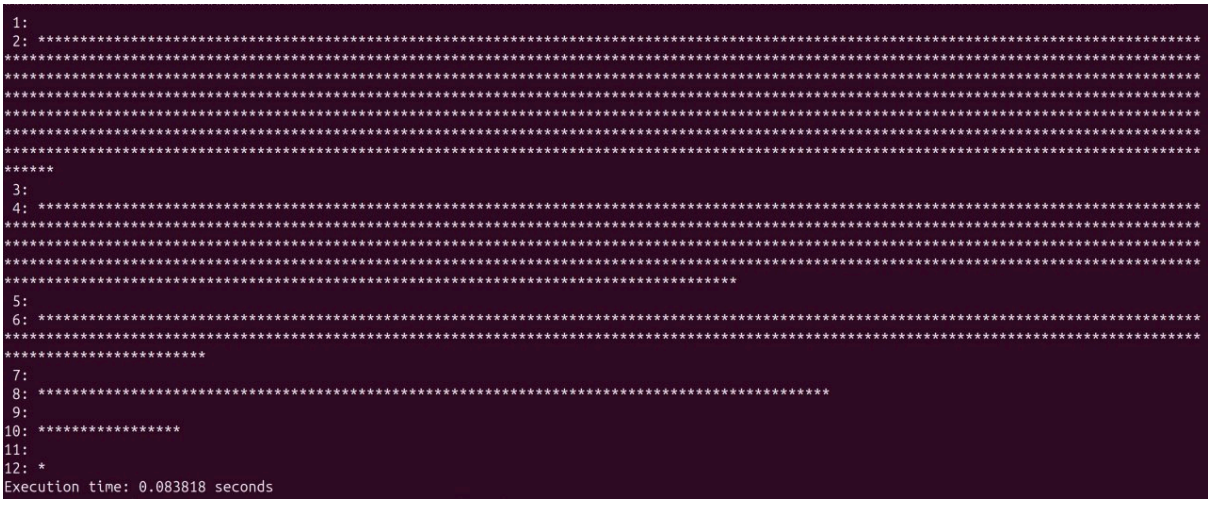
تعداد نمونه	۵۰۰۰	۵۰۰۰۰	۵۰۰۰۰۰
زمان اجرا	0.001219 seconds	0.012186 seconds	0.083818

کد اول با ۵۰۰۰:



کد اول با ۵۰۰۰۰:

کد اول با ۵۰۰۰۰۰:



کد قسمت دوم با ۵۰۰۰:

کد قسمت دوم با ۵۰۰۰۰:

کد قسمت سوم با ۵۰۰۰۰۰:

```
 1:
 2: ********************************************************************************************************************************************************
********************************************************************************************************************************************************
********************************************************************************************************************************************************
********************************************************************************************************************************************************
********************************************************************************************************************************************************
********************************************************************************************************************************************************
********************************************************************************************************************************************************
*********
 3:
 4: ********************************************************************************************************************************************************
********************************************************************************************************************************************************
********************************************************************************************************************************************************
********************************************************************************************************************************************************
********************************************************************************
 5:
 6: ********************************************************************************************************************************************************
********************************************************************************************************************************************************
**********************
 7:
 8: ******************************************************************************************************
 9:
10: ******************
11:
12: **
Execution time: 0.000641 seconds
```