**Laborator 05**

**Exerciții**

**Implementați și măsurați timpii de execuție pentru setul de coduri de mai jos.**

* **Rezultatele vor fi notate in timpi.csv**
* **Vor fi necesare alocări și inițializări (se poate folosi random).**
* **Nu faceți printări decât pentru validare, când măsurați timpii de execuție scoateți printările.**
* **Toate matriciile și vectorii trebuie alocați cu malloc().**
* **Se vor folosi valori lui N puteri a lui 10. Adică 10, 100, 1000 și tot așa.**
* **Nu toate valorile vor putea fi testate pe toate programele (unele vor dura mult prea mult).**
* **Neapărat compilați cu debug (Launch are activată optimizare -O3 și va șterge orice consideră cod inutil).**

**Pentru a măsura timpul de execuție (se ia valoarea real):**

**time ./executabil N**

1. (**factorial.c)**

long factorial(int n)

{

    if (n <= 1)

        return 1;

    else

        return n \* factorial(n - 1);

}

1. (**matrixMultiply.c**)

for (int i = 0; i < N; i++)

    for (int j = 0; j < N; j++)

        for (int k = 0; k < N; k++)

            c[i][j] += a[i][k] \* b[k][j];

1. (**bubbleSort.c)**

for (int i = 0; i < N - 1; i++)

    for (int j = 0; j < N - i - 1; j++)

        if (a[j] > a[j + 1])

        {

            aux = a[j];

            a[j] = a[j + 1];

            a[j + 1] = aux;

        }

1. (**binarySearch.c)**

int binarySearch(int arr[], int l, int r, int x)

{

    if (r >= l)

    {

        int mid = l + (r - l) / 2;

        if (arr[mid] == x)

            return mid;

        if (arr[mid] > x)

            return binarySearch(arr, l, mid - 1, x);

        return binarySearch(arr, mid + 1, r, x);

    }

    return -1;

}

1. (**dirty.c)**

int i = N;

while (i > 0)

{

    int Sum = 0;

    int j;

    for (j = 0; j < i; j++)

        Sum++;

    printf("%i\n", Sum);

    i--;

}

1. (**strangeCount.c)**

int counter = 0;

for (int i = 0; i < N; i++)

    for (int j = 1; j < N; j \*= 2)

        counter++;

printf("%i\n", counter);

1. (**strangeCount3.c)**

int counter = 0;

for (int i = 0; i < N; i++)

    for (int j = 1; j < pow(i, 2); j++)

        for (int k = 0; k < j; k++)

            counter++;

printf("%i\n", counter);

1. (**strangeCount4.c)**

int counter = 0;

int i = 1;

while (i <= N)

{

    i = i \* 2;

    for (int j = 1; j < log10(i) / log10(2); j++)

        for (int k = 1; k <= j; k++)

            counter++;

}

printf("%i\n", counter);