Laborator 03

Exerciții

Calculați complexitatea următorilor algoritmi:

1. (**strlen.c**)

```c
int strlen(const char *s)
{
    int i;
    for (i = 0; s[i] != '\0'; i++)
        ;
    return i;
}
```

2. (**matrixMultiply.c**)

```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];
```

3. (**isNumberEven.c**)

```c
int isNumberEven(int no)
{
    return !(no % 2);
}
```

4. (**doubleCount.c**)

```c
int count = 0;
for (int i = 0; i < N; i++)
    count++;
for (int j = 0; j < M; j++)
    count++;
```

5. (**bigAndSmall.c**)

```c
int count = 0;
for (int i = 0; i < N; i++)
    for (int j = 0; j < N; j++)
        count++;
for (int k = 0; k < N; k++)
    count++;
```

6. **(strangeSum.c)**

```c
int y = 0;
int j = N;
while (j >= 1)
{
    int k = 0;
    while (k < N * N)
    {
        y = y + j - k;
        k++;
    }
    j--;
}
printf("%i\n", y);
```

7. **(bubbleSort.c)**

```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;
        }
```

8. **(dirty.c)**

```c
int i = N;
while (i > 0)
{
    int Sum = 0;
    int j;
    for (j = 0; j < i; j++)
        Sum++;

    printf("%i\n", Sum);
    i--;
}
```

9. **(strangeCounter.c)**

```c
int counter = 0;
for (int i = 0; i < N; i++)
    for (int j = 1; j < N; j *= 2)
        counter++;
printf("%i\n", counter);
```

10. **(strangeCounter2.c)**

```
int counter = 0;
for (int i = 0; i < N; i++)
    for (int j = 1; j < i; j++)
        for (int k = j; k < i + j; k++)
            counter++;
printf("%i\n", counter);
```

11. **(strangeCounter3.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);
```

12. **(strangeCounter4.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);
```

13. **(factorial.c)**

```
long factorial(int n)
{
    if (n <= 1)
        return 1;
    else
        return n * factorial(n - 1);
}
```

14. (**binarySearch.c**)

```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;
}
```