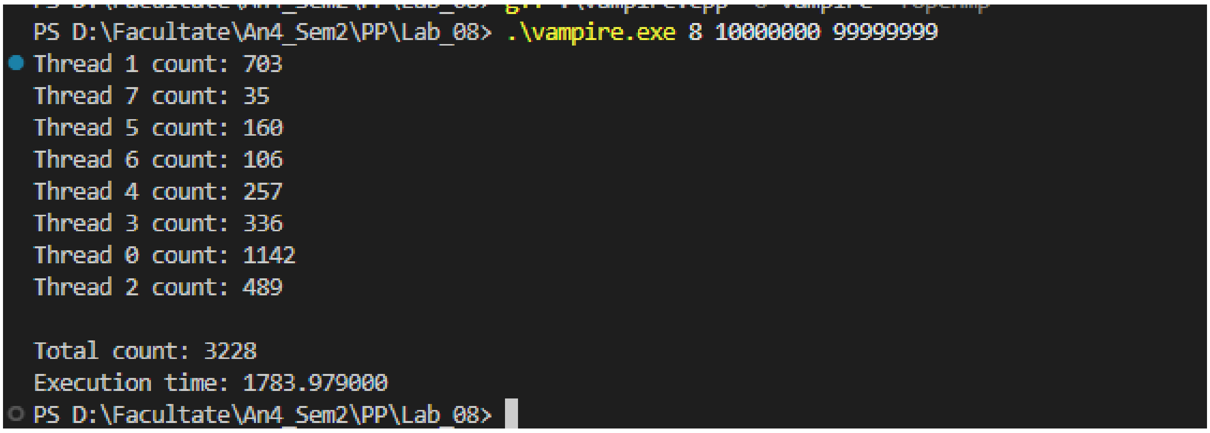
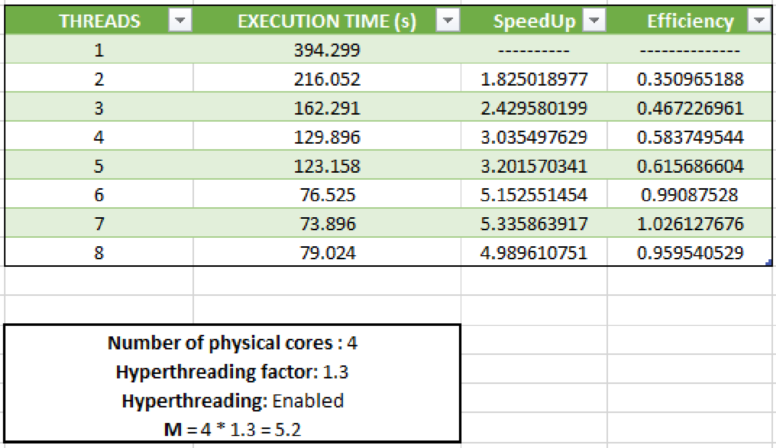


```
PS D:\Facultate\An4_Sem2\PP\Lab_08> .\vampire.exe 8 10000000 99999999
Thread 1 count: 703
Thread 7 count: 35
Thread 5 count: 160
Thread 6 count: 106
Thread 4 count: 257
Thread 3 count: 336
Thread 0 count: 1142
Thread 2 count: 489

Total count: 3228
Execution time: 1783.979000
PS D:\Facultate\An4_Sem2\PP\Lab_08>
```

Pentru intervalul 10.000.000 – 99.999.999: 3228 numere vampirice

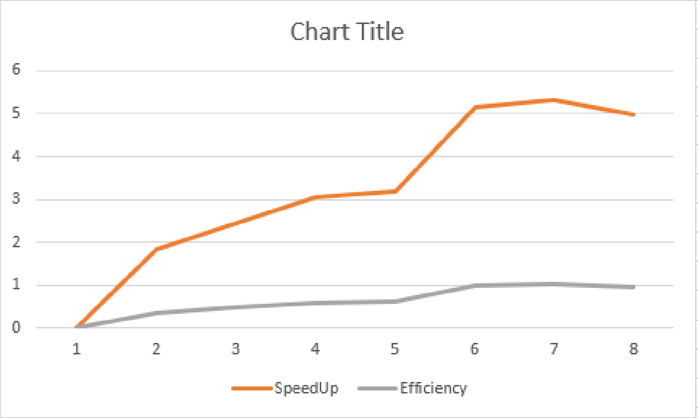
Comanda: vampire.exe <no_threads> <interval_start> <interval_end>

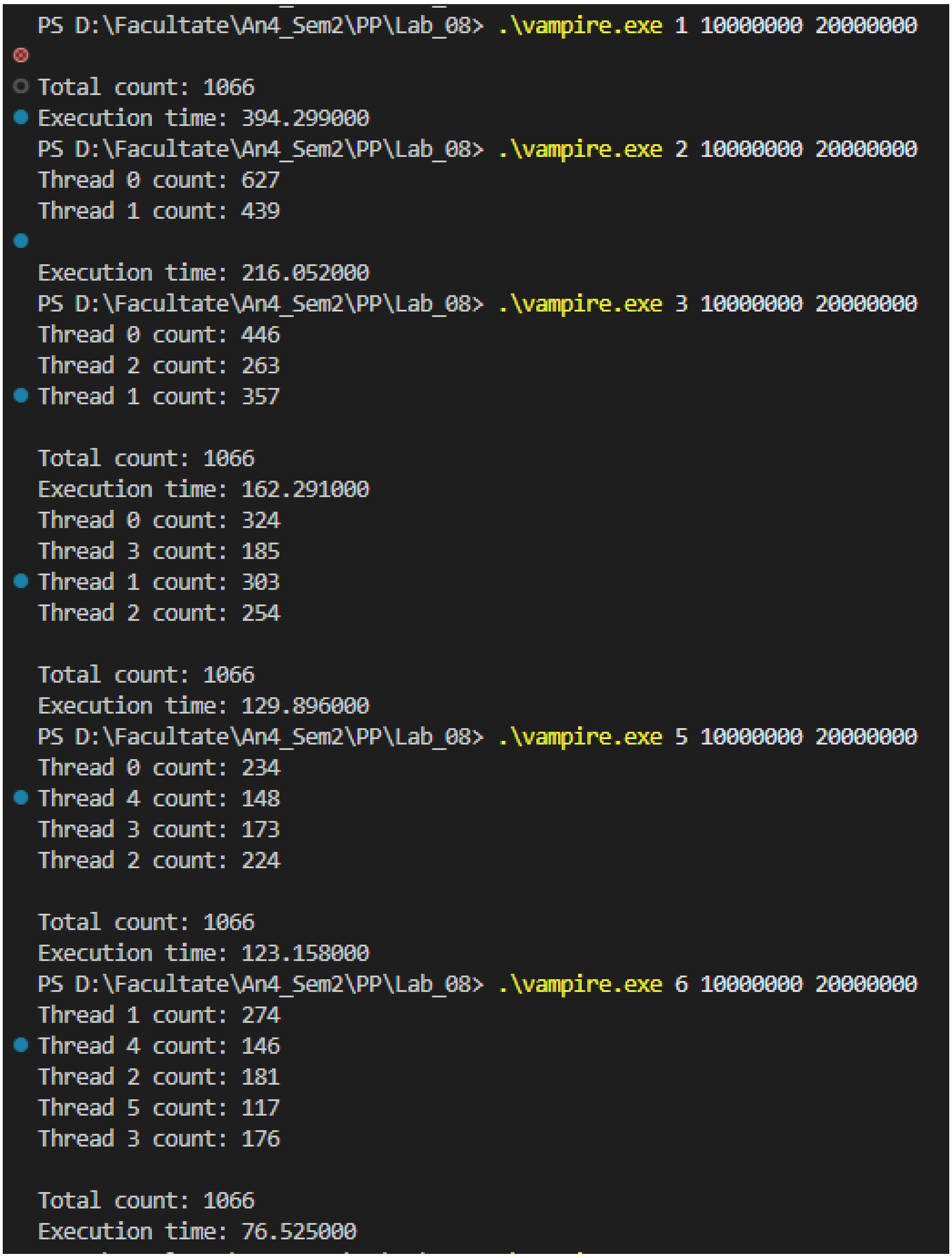
TABEL:



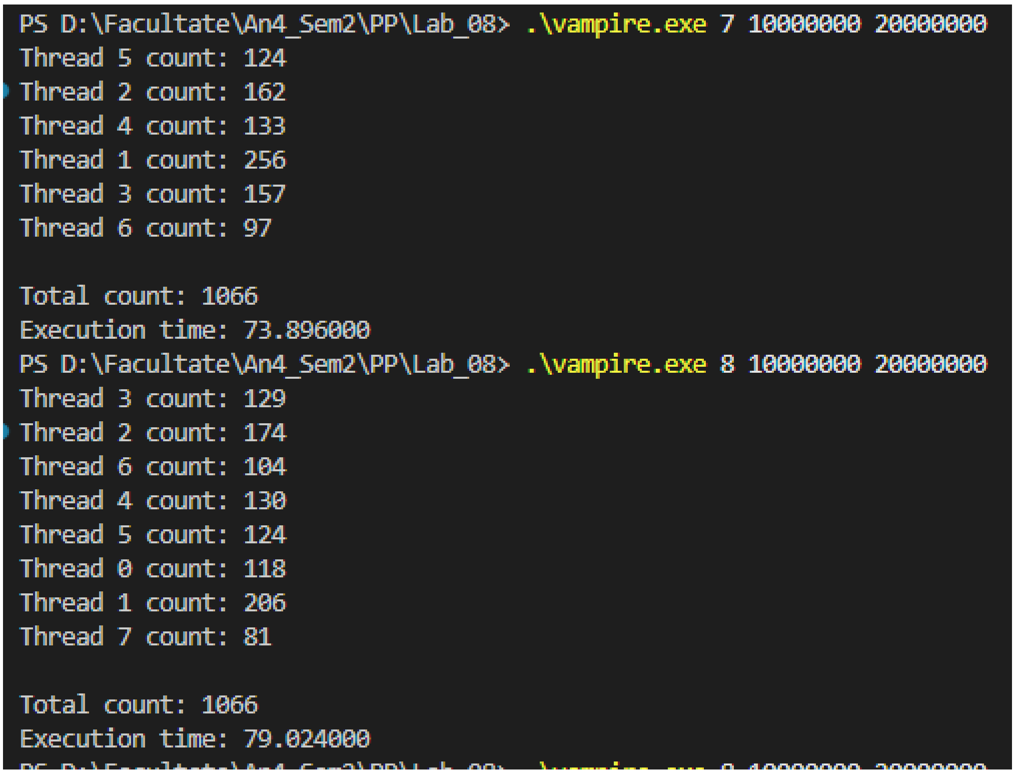
THREADS	EXECUTION TIME (s)	SpeedUp	Efficiency
1	394.299	----------	--------------
2	216.052	1.825018977	0.350965188
3	162.291	2.429580199	0.467226961
4	129.896	3.035497629	0.583749544
5	123.158	3.201570341	0.615686604
6	76.525	5.152551454	0.99087528
7	73.896	5.335863917	1.026127676
8	79.024	4.989610751	0.959540529

Number of physical cores : 4
Hyperthreading factor: 1.3
Hyperthreading: Enabled
M = 4 * 1.3 = 5.2



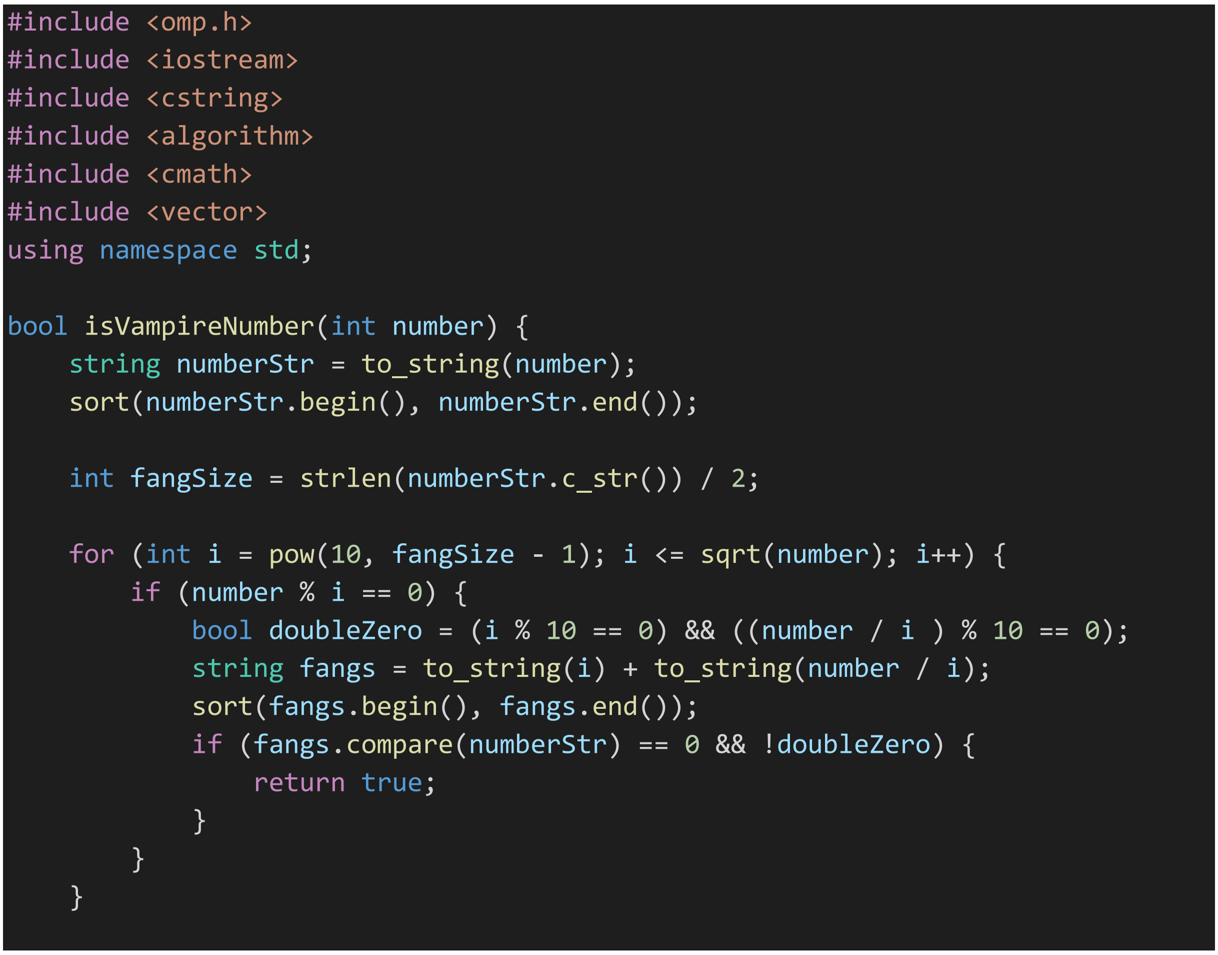


```
PS D:\Facultate\An4_Sem2\PP\Lab_08> .\vampire.exe 1 10000000 20000000

Total count: 1066
Execution time: 394.299000
PS D:\Facultate\An4_Sem2\PP\Lab_08> .\vampire.exe 2 10000000 20000000
Thread 0 count: 627
Thread 1 count: 439

Execution time: 216.052000
PS D:\Facultate\An4_Sem2\PP\Lab_08> .\vampire.exe 3 10000000 20000000
Thread 0 count: 446
Thread 2 count: 263
Thread 1 count: 357

Total count: 1066
Execution time: 162.291000
Thread 0 count: 324
Thread 3 count: 185
Thread 1 count: 303
Thread 2 count: 254

Total count: 1066
Execution time: 129.896000
PS D:\Facultate\An4_Sem2\PP\Lab_08> .\vampire.exe 5 10000000 20000000
Thread 0 count: 234
Thread 4 count: 148
Thread 3 count: 173
Thread 2 count: 224

Total count: 1066
Execution time: 123.158000
PS D:\Facultate\An4_Sem2\PP\Lab_08> .\vampire.exe 6 10000000 20000000
Thread 1 count: 274
Thread 4 count: 146
Thread 2 count: 181
Thread 5 count: 117
Thread 3 count: 176

Total count: 1066
Execution time: 76.525000
```

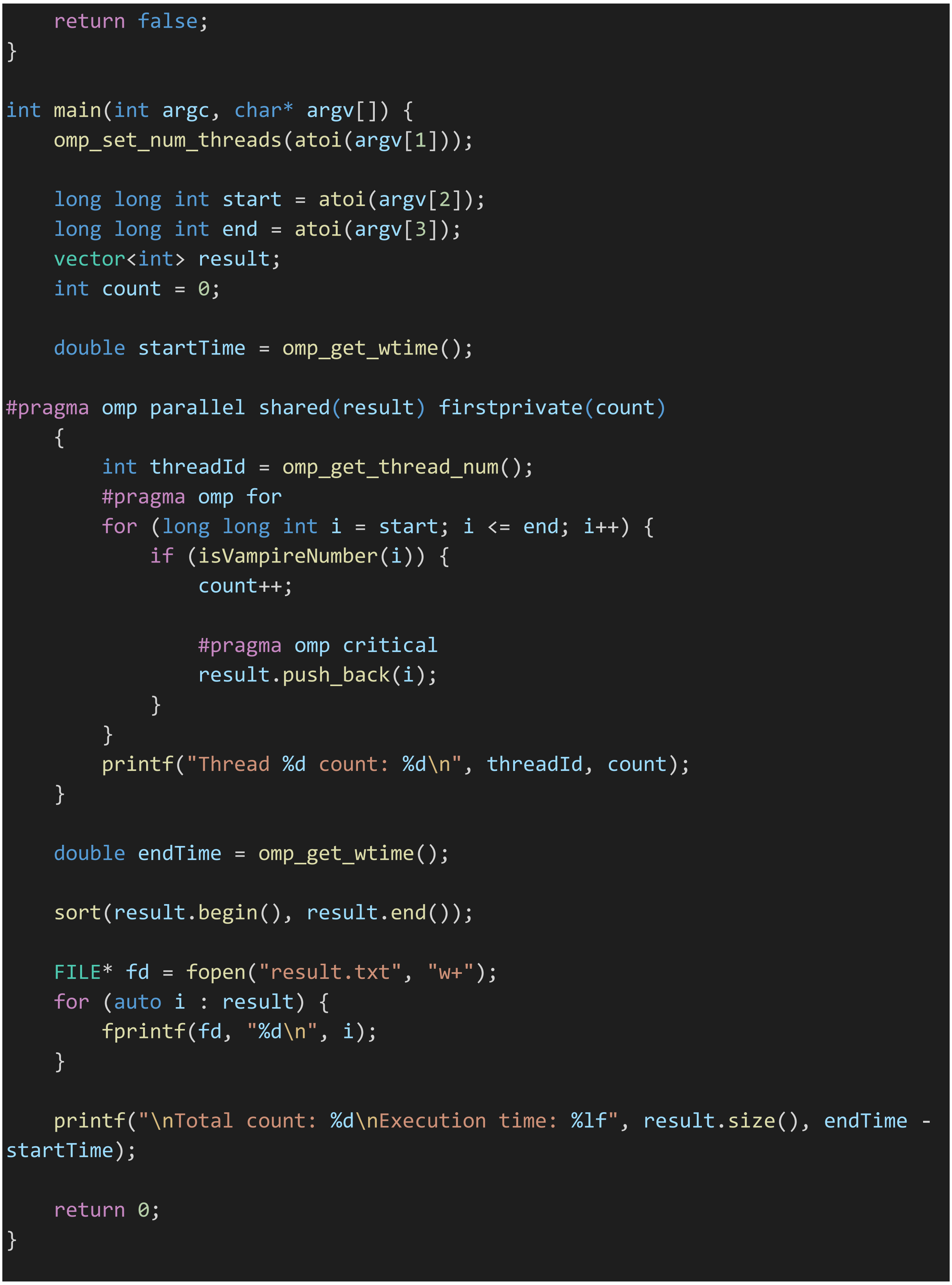


```
PS D:\Facultate\An4_Sem2\PP\Lab_08> .\vampire.exe 7 10000000 20000000
Thread 5 count: 124
Thread 2 count: 162
Thread 4 count: 133
Thread 1 count: 256
Thread 3 count: 157
Thread 6 count: 97

Total count: 1066
Execution time: 73.896000
PS D:\Facultate\An4_Sem2\PP\Lab_08> .\vampire.exe 8 10000000 20000000
Thread 3 count: 129
Thread 2 count: 174
Thread 6 count: 104
Thread 4 count: 130
Thread 5 count: 124
Thread 0 count: 118
Thread 1 count: 206
Thread 7 count: 81

Total count: 1066
Execution time: 79.024000
PS D:\Facultate\An4_Sem2\PP\Lab_08> .\vampire.exe 9 10000000 20000000
```

COD:



```
#include <omp.h>
#include <iostream>
#include <cstring>
#include <algorithm>
#include <cmath>
#include <vector>
using namespace std;

bool isVampireNumber(int number) {
    string numberStr = to_string(number);
    sort(numberStr.begin(), numberStr.end());

    int fangSize = strlen(numberStr.c_str()) / 2;

    for (int i = pow(10, fangSize - 1); i <= sqrt(number); i++) {
        if (number % i == 0) {
            bool doubleZero = (i % 10 == 0) && ((number / i ) % 10 == 0);
            string fangs = to_string(i) + to_string(number / i);
            sort(fangs.begin(), fangs.end());
            if (fangs.compare(numberStr) == 0 && !doubleZero) {
                return true;
            }
        }
    }
```



```
    return false;
}

int main(int argc, char* argv[]) {
    omp_set_num_threads(atoi(argv[1]));

    long long int start = atoi(argv[2]);
    long long int end = atoi(argv[3]);
    vector<int> result;
    int count = 0;

    double startTime = omp_get_wtime();

#pragma omp parallel shared(result) firstprivate(count)
    {
        int threadId = omp_get_thread_num();
        #pragma omp for
        for (long long int i = start; i <= end; i++) {
            if (isVampireNumber(i)) {
                count++;

                #pragma omp critical
                result.push_back(i);
            }
        }
        printf("Thread %d count: %d\n", threadId, count);
    }

    double endTime = omp_get_wtime();

    sort(result.begin(), result.end());

    FILE* fd = fopen("result.txt", "w+");
    for (auto i : result) {
        fprintf(fd, "%d\n", i);
    }

    printf("\nTotal count: %d\nExecution time: %lf", result.size(), endTime -
startTime);

    return 0;
}
```