```
#include <algorithm>
#include <iostream>
#include <thread>
#include <string>
#include <fstream>
#include <vector>
#include <omp.h>

using namespace std;

vector<vector<string>> arr, help;

inline bool space(char c) {
	return isspace(c);
}

inline bool notspace(char c) {
	return !isspace(c);
}

vector<string> split(const string& s) {
	typedef string::const_iterator iter;
	vector<string> ret;
	iter i = s.begin();
	while (i != s.end()) {
		i = find_if(i, s.end(), notspace);
		iter j = find_if(i, s.end(), space);
		if (i != s.end()) {
			ret.push_back(string(i, j));
			i = j;
		}
	}
```

```
        return ret;
}

void mult(int first, int place) {
        int f = first;
        int p = place;
        int k = 0;
        unsigned long long g = arr[f].size() * arr[f + 1].size();
        help[p].resize(g);
        for (int i = 0; i < arr[first].size(); i++) {
                for (int j = 0; j < arr[f + 1].size(); j++) {
                        help[p][k] = arr[f][i] + ", " + arr[f + 1][j];
                        k++;
                }
        }
}

int main()
{
        setlocale(LC_ALL, "RUS");
        ifstream file;
        ofstream ofile;
        string in_path, out_path, line, num;
        int  count_of_threads, count_of_plenty, count_of_numbers, size;
        vector<string> current_line, output_strings;
        char str[100];

        cout << "Введите число множеств" << endl;
        cin >> count_of_plenty;
        cout << "Введите количесвто чисел" << endl;
        cin >> count_of_numbers;
        cout << "Введите число потоков" << endl;
```

```
cin >> count_of_threads;
in_path = "Manys.txt";
out_path = "ans.txt";
arr.resize(count_of_plenty);



file.open(in_path);
if (!file.is_open()) {
	cout << "Could not open the file!";
	system("pause");
	exit(EXIT_FAILURE);
}

ofile.open(out_path);
if (!ofile.is_open()) {
	cout << "Could not open the file!";
	system("pause");
	exit(EXIT_FAILURE);
}

for (int i = 0; i < count_of_plenty; i++) {
	arr[i].resize(count_of_numbers);
	file.getline(str, 100);
	line = string(str);
	current_line = split(line);
	for (int j = 0; j < count_of_numbers; j++) {
		num = current_line[j];
		arr[i][j] = num;
	}
}

ofile << "{";
```

```
while (arr.size() != 1)
{
	help = arr;
	if (arr.size() % 2 == 0) {
		int place = 0;
		int first = 0;
		size = arr.size() / 2;
		#pragma omp parallel num_threads(count_of_threads)
		#pragma omp parallel while
		while (place != size)
		{
			mult(first, place);
			cout << "Thread: " << omp_get_thread_num() << endl;
			place += 1;
			first += 2;
		}
	}
	else {
		int place = 1;
		int first = 1;
		size = arr.size() / 2 + 1;
		#pragma omp parallel num_threads(count_of_threads)
		#pragma omp parallel while
		while (place != size)
		{
			mult(first, place);
			cout << "Thread: " << omp_get_thread_num() << endl;
			place += 1;
			first += 2;
		}
	}
	arr = help;
```

```
                arr.resize(size);
        }
        for (int i = 0; i < arr[0].size(); i++) {
                ofile << "(" + arr[0][i] + "); ";
        }
        ofile << "}";
}
```