מחשוב מקבילי ומבוזר

תרגיל #1

The purpose of this exercise is to implement a simple application with **Dynamic** and **Static** Task Pool management.

Parallelize the following code

```
#define FILE_NAME "points.txt"

// This function simulates heavy computations,
// its run time depends on x, y and param values
// DO NOT change this function!!

double heavy(double x, double y, int param) {
	double center[2] = { 0.4, 0.2 };
	int i, loop, size = 1, coeff = 10000;
	double sum = 0, dx, dy, radius = 0.2*size;
	int longLoop = 1000, shortLoop = 1;
	double pi = 3.14;
	dx = (x - center[0]) * size;
	dy = (y - center[1]) * size;
	loop = (sqrt(dx * dx + dy * dy) < radius) ? longLoop : shortLoop;

	for (i = 1; i < loop * coeff; i++)
		sum += cos(2*pi * dy * dx + 0.1) * sin(exp(10*cos(pi* dx))) / i;

	return  sum;
}
```

```
// Reads data from from the file and allocates the array of points
// The first line contains a parameter
// The second line contains a number of points defined.
// Following lines contain two doubles each - point coordinates x, y



double* readFromFile(const char* fileName, int* numberOfPoints, int *param) {
	FILE* fp;
	double* points;

	// Open file for reading points
	if ((fp = fopen(fileName, "r")) == 0) {
		printf("cannot open file %s for reading\n", fileName);
		exit(0);
	}

	// Param
	fscanf(fp, "%d", param);

	// Number of points
	fscanf(fp, "%d", numberOfPoints);

	// Allocate array of points end Read data from the file
	points = (double*)malloc(2 * *numberOfPoints * sizeof(double));
	if (points == NULL) {
		printf("Problem to allocate memory\n");
		exit(0);
	}
	for (int i = 0; i < *numberOfPoints; i++) {
		fscanf(fp, "%le %le", &points[2 * i], &points[2 * i + 1]);
	}

	fclose(fp);

	return points;
}
```

```
int main(int argc, char* argv[]) {
	double answer = 0;
	int numberOfPoints = 10;
	double *points, x, y;
	int param;

	// Read points from the file
	points = readFromFile(FILE_NAME, &numberOfPoints, &param);

	// Find maximum value of heavy calculated for each point
	x = points[0];
	y = points[1];
	answer = heavy(x, y, param);

	// Perform heavy sequential computation
	for (int i = 1; i < numberOfPoints; i++) {
		x = points[2 * i];
		y = points[2 * i + 1];
		answer = fmax(answer, heavy(x, y, param));
	}

	printf("answer = %e\n", answer);

	return 1;
}
```

Requirements:

1. Implement two approaches to parallelize the code:
   a. Use **Static Task Pool** approach to solve the problem
   b. Implement **Dynamic Task Poo**l Approach for parallel solution

2. Run, measure execution time, explain the results. The table with the time measurement is to be placed in the separate Word file named **results.doc** in the root directory of the solution.

3. No changes to function **heavy**() are allowed. It is considered as a “black box”, meaning that your solution is not based on understanding what kind of computation is made and how long it may run for specific parameters x and y.

Solution type	Number of Slaves	Execution time	Explain the result
Sequential Solution	N/A		
Static Task Pool	2		
Static Task Pool	4		
Static Task Pool	10		
Dynamic Task Pool	2		
Dynamic Task Pool	4		
Dynamic Task Pool	10		

Grading Policy:

- **10 points** for code quality:
  a. The code must be divided into small functions (not more than 40 lines of code).
  b. Use meaningful names for variables, functions, files, constants.
  c. Place enough comments to understand the code
  d. No unused lines of code. Don't repeat the code – use functions!
  e. Write README.TXT file if special instructions are needed to run the solution. The file must be in the root folder of the solution.
- **70 points** – for proper implementation of the requirements.

- **20 points** – for final results explanation and for time measurement.

Important:

- The Homework has to be tested under Ubuntu OS in VLAB with compilation and run from Terminal.

- Perform time measurement on VLAB. Make few runs and use an average value.

- The Homework must be delivered in time. No delay will be accepted. It may be performed in pairs. *Only one member of pair* submits the solution through the Moodle.

- The whole solution must be zipped and named as

111111111_222222222.zip

Where **111111111** is ID of the one student and **222222222** is ID of another student

בהצלחה!