BİL 214 – System Programming

Homework #8

Assigned on 22.11.2022 – Due on 29.11.2022

- Submit one C source file per question.
- If a source file fails to compile with the gcc compiler, you get zero credits for that question.
- Make sure your submission file names are formatted as:
  FirstName_LastName_StudentID_HW#_Q#.c
  For example: Toygar_Akgun_123456789_HW8_Q1.c
  Toygar_Akgun_123456789_HW8_Q2.c …

1. [50 points] Write a C program that calculates the 50th Fibonacci number using the inefficient recursive implementation we discussed in the classroom. This implementation is available in the code samples shared on Piazza. You can just grab that code from there. Once you have that read, write (or copy/paste) additional timing code to measure the total execution time it takes to calculate the 50th Fibonacci number on your system.

   As we discussed in the class, there is a way of using two threads to accelerate this implementation about 1.5 - 1.6 times. Use POSIX threads to implement that idea and time your multi-threaded implementation. Print both (single threaded, multi-threaded) execution times on the terminal with clear information messages.

2. [25 points] A tensor is a generalization of vectors and matrices and can be interpreted as a multi-dimensional array.

   Write a C program that allocates an integer tensor with dimensions MxNxQ. Assume that M, N and Q can be in the range [1, 1000]. Ask the user to input these dimensions via terminal. Use an (int ***) type pointer and a triple nested for loop for your dynamic memory allocation. Then randomly initialize your tensor elements to integers in the range [0, 10]. Once your tensor is allocated and initialized, add a random integer in the range [-5, 5] to all tensor elements.

3. [25 points] Write a C program that does the same thing as in Question (2), but this time use an (int *) type pointer and a single malloc to get your dynamic memory allocation, initializations and additions done.

4. [0 points – **You do not need to deliver an answer for this question**] Time your codes for Questions (2) and (3). You might want to repeat the overall execution (allocate, rand init and rand add) say 100 or 1000 times and average over these measurements to make them more accurate. Are the timings same or different? Try the same experiment with turning compiler optimizations ON/OFF with -O3 and -O0. Are there any changes?