HIGH PERFORMANCE COMPUTING AND PROGRAMMING
UPPSALA UNIVERSITY
SPRING 2016
EXAMINATION

1. Instructions

Your task is to write a program that creates and manipulates data in the way described below.

- The program takes the integer N as a command-line parameter. It is callable by the command `./exam 100`.
- Create an array of star structs of length N. Initialize each star randomly according to the comments in the struct definition (`common.h`).
- Sort the stars according to distance from the origin with your preferred sort method.
- Create a `float_t` matrix of size NxN. Element (i, j) of the matrix is equal to the distance between star i and star j plus the value of `starfunc()` for stars i and j.
- For each interior element of the matrix, take the absolute value of the difference of the element and each element in its von Neumann neighborhood (i.e. elements up, down, left, and right). The mean of these four differences is a result that is tallied in a histogram.
- The histogram has 10 bins, split equally between the minimum and maximum results. Each bin is an integer that represents the number of elements with a range of results.

1.1. **Skeleton.** You are provided with a code skeleton and some utility functions for displaying the star array, matrix, and histogram.

Your first task is to create a working program from the skeleton structure which generates correct results. All code must be your own, including functions for e.g. sorting. Use library functions only for the most basic operations like `sqrt()`.

Reference input and output data for N=7 that can be used to verify correctness is available in `ref_input.c` and `ref_output.txt`.

1.2. **Optimization.** Profile your code and optimize in any way you can. *Save each working version.* You may change the structure of the code, but the initialization of the star structs and the final histogram must remain the same, so result correctness can be verified by comparing to the histogram given by the unoptimized code. Focus on making your code efficient for large N e.g. N=5000 and upwards.

Write a 1-2 page summary of your optimizations where you show their effectiveness. If an attempted optimization does not work and you know why it does not work, include this in your text. *Hint: figures are a good way of presenting results concisely.*

Date: May 18, 2016.

2. Submission and Examination

The mandatory exam will take place on the *3rd of June, 09:15-16:00* in computer lab *2507*.

On the day of the exam, you will be interviewed in turn, by the computer. This is your opportunity to show us what you have learned!

If you want to take the exam, we ask you to submit the code and report under "Examinatory miniproject" in the Student Portal by the *31st of May, 23:59*. The order of the submission will determine the order of the interviews, times will be announced by email in the morning of the 1st of June.

Late submissions are unacceptable and you will have to wait until August to try again.

Your submission should consist of a single `.tar.gz` file including your code and report files:

- Runnable unoptimized code (including a *makefile*)
- Runnable optimized code (including a *makefile*)
- Report in pdf format

Important: the exam work should be carried out *individually*. You can talk to other students, but don't do it in front of a computer and don't write anything down. The bottom line is that we have to be convinced that the handed-in code and report are done by you and understood by you.

Also important: Bring an ID card, passport or equivalent, on the exam day so that we can verify that you are who you say you are.

3. Grading

You will be graded according to a (secret) rubric based on the complete course contents. The grade requirements are as follows:

- **3** You must demonstrate an understanding of the fundamentals of the course. Your code works correctly and you have made a reasonable attempt at optimizing the runtime of the code.
- **4** You have exhaustively tried to optimize everything, even code that isn't performance critical. You're able to reason about the performance of your code at a high level and know why certain optimizations worked/didn't work.
- **5** To earn the highest grade, you must formulate a new function/routine that allows you to demonstrate at least one optimization technique that you were not able to use (i.e. benefit from) in your final solution. In other words, develop a little code that performs some operation on one of the data structures in your solution and devise it in such a way that you can showcase an optimization technique not seen elsewhere in your code.

4. Questions

If there are any questions, email: `elias.rudberg@it.uu.se`