**Lab 4:**

You may work in a group of up to 2 people. If you choose to do this, ensure that you put the names of all team members into your lab4.md file. Working together can help you understand the content of the videos. You are all expected to watch the video and be part of the discussion. **Each member must submit a copy of the work, but the answers for part A can be the same. Reflection needs to be independently done.**

1. Watch the following video: [Sorting Algorithms: Speed Is Found In The Minds of People - Andrei Alexandrescu - CppCon 2019](https://www.youtube.com/watch?v=FJJTYQYB1JQ)
2. Answer the questions in the next section. You can use the internet to help you learn unfamiliar ideas, but you will need to link and cite your sources of information. A link may be helpful for you at the bottom of this lab.
3. Write a reflection (this part is individual)

**Part A: Questions about the video**

Do not forget to add all group members' names (and email addresses) to the top of the file if you are working in a group.

1. What sorting algorithm was the speaker trying to improve?
   * + The speaker trying to improve “std::sort”, which is generally an implementation of introverted alignments (in certain cases, fast alignments that switch to hip-sorts or insertion alignments).
2. At what partition size does VS perform a simpler sort algorithm instead of continuing to partition?
   * + 32 elements or fewer
3. At what partition size does GNU perform a simpler sort algorithm instead of continuing to partition?
   * + Less than or equal to 16 elements
4. Regular insertion sort does a linear search backwards from end of array for correct spot to insert. According to the speaker, why does switching to a binary search not improve performance?
   * + Modern CPU branch predictors predict simple linear tests so well that binary searches can increase branch misprediction and slow down.
5. Describe what is meant by branch prediction. (this may require further research)
   * + "For CPU to pre-predict the true/false of the condition and fill the pipeline."
6. What is meant by the term **informational entropy**? (this may require further research)
   * + Randomness of the elements.
7. Speaker suggests the following algorithm:
   * make\_heap()
   * unguarded\_insertion\_sort()

He suggests that by doing make\_heap() first, you can do something called unguarded\_insertion\_sort(). Please explain what unguarded\_insertion\_sort() is and why it is faster than regular insertion sort. How does performing make\_heap() allow you to do this?

* + - Make\_heap() puts the minimum value at the front, allowing you to loop the inside loop of the insert alignment without the boundary check.

1. The speaker talks about incorporating your conditionals into your arithmetic. What does this mean? Provide an example from the video and explain how the conditional is avoided.
   * + sum += (x < y) \* diff; the part where the branch is changed to an operation (0 or 1)
2. The speaker talks about a bug in gnu's implementation. Describe the circumstances of this bug.
   * + The part that pointed to the overflow problem of low + high in the alignment implementation of GNU lib std c++.
3. The speaker shows several graphs about what happens as the threshold increases using his new algorithm. The metric of comparison is increased, and the metric of moves is increased, but time drops... What metric does the author think is missing? Describe the missing metric he speaks about in the video. What is the metric measuring?
   * + The speaker said the missing indicators represent distance, or how far the elements travel in memory. He explained that very short distances (close to each other) of elements improve cache behavior and performance. Measuring travel distances is more important in real speed than just comparing or swapping.
4. What does the speaker mean by fast code is left-leaning?
   * + He means that for hot code paths, conditions and data must be arranged so that the most likely outcome is on the left path. This reduces branch mispredictions and enables more linear and cache-friendly execution.
5. What does the speaker mean by not mixing hot and cold code?
   * + It is also true that we do not contaminate cache and branch predictor by separating exception handling or rare paths (cold codes) from areas where core performance is important (hot codes).

**Part B: Reflection**

This part must be individually done.

1. What did you/your team find most challenging to understand in the video?
   * + Branch prediction and its effect on the alignment speed were difficult.
2. What is the most surprising thing you learned that you did not know before?
   * + It was most surprising that binary navigation can be rather slow in insertion alignment.
3. Has the video given you ideas on how you can write better/faster code? If yes, explain what you plan to change when writing code in the future. If not, explain why not.
   * + In areas where performance should be considered important, we are not just choosing algorithms, but we are also considering compiler optimization or cache efficiency.

**References:**

You may find these articles on branching and cache useful:

<https://en.algorithmica.org/hpc/>

**Submitting your lab**

**Please upload all your answers in a single file in GitHub and Blackboard Ultra.**