Overview: The content your discussion group creates in this document will be synthesized in a blog posting for https://bssw.io

Instructions:
1. Pick one person in your discussion group to create a new copy of this Google Doc
2. Make a copy of this template in a new Google Doc (the person from step 1)
3. Share the edit link to the document in step 2 with others (copy and paste into Zoom chat)
4. Co-edit the document: Can have one lead writer with others modifying, or another approach
5. Send the document to Mike Heroux at the end of the session by email (mheroux@csbsju.edu)

Add group member names for anyone who wants attribution in the blog post:
1. Todd Munson, Argonne National Laboratory, github: tmunson
2. Anshu Dubey, Argonne National Laboratory, github: adubey64
3. Sam Yates, Swiss National Supercomputing Centre; GitHub ID: halfflat
4. Barry Smith, Argonne National Laboratory
5. Sarah Knepper, Intel Corporation; GitHub ID: sknepper
6. Ulrike Meier Yang, Lawrence Livermore National Laboratory; github id: ulrikeyang

Add a paragraph describing each person's software team background and experience. Emphasize experiences that inform your opinions about challenges, and how to improve software teams. Can add paragraphs in parallel :)
- (Anshu) Lead for FLASH/Flash-X (a multiphysics multi-domain scientific software) development.
- (Todd) I work with the Toolkit for Advanced Optimization, which is part of the PETSc project, and manage the ECP Software Ecosystem and Deployment portfolio
- (Ulrike) I work with the xSDK and the hypre project
- (Sam) Technical lead for the Arbor simulator project (morphologically detailed neuron and neuron network simulator) — small, distributed team.
- (Sarah) Manager on the Intel oneAPI Math Kernel Library (oneMKL) team, closed-source product. Multiple developers on multiple teams contributing to a really large, complex, long-standing code source.

Discuss as a group the most important challenges you see facing scientific software teams. Try to refrain from solving the challenges (that's tomorrow and the next day). Summarize discussion in outline form.
- Diffuse teams and how to spontaneously self organize
- Funding to sustain and develop the software

    - Different groups get different funding at different times
    - Shifting priorities and center of gravity that are funding related
    - Shifting priorities by funding agencies
- Career paths for software developers
- Attracting and retaining staff
- Staff are unable to focus, since they are spread out over too many tasks due to small pieces of funding here and there
- Knowledge sharing within and between teams
    - Many teams that contribute
    - Knowledge silos may be present (cooperation vs. competition)
    - Some may not want to share about unfinished products
    - Funding can create a barrier for knowledge sharing (e.g. don't share the secret research sauce until you get the funds)
- Onboarding of new staff; offboarding of staff
    - Keeping documents up to date
- Transition from a local group to a distributed group, especially as the team grows
- Impact of work from anywhere becoming more prevalent
- Key personnel leaving; succession planning
- Ownership / stickiness when you provide something
- Abandonment of contributed code or having the "core" take over the contributed code
- How is the value perceived among the users? (e.g., technicians or partners)
- How is the value perceived by your organization?
- Visibility and citation of software
- Tools used -- homegrown vs. pre-existing solutions
    - Frameworks, built into code
- Human resources get in the way or limit our ability to maintain an effective team
- So do DOE's concerns about "foreign talent"
- Tracking the entire project as it evolves spontaneously with many contributions in various areas that are not coordinated or centrally managed. No one has the time to read all MR, bug reports and issues so how do you condense the information into a hierarchical "big picture" report
- Dependencies, interoperability
- Tangential contributions
    - Long-term maintainability
- Maintainability of all the "performance portability" layers and backends
    - Maintainability of support for all the accelerators
- Mismatch between state of code and the documentation
    - Time required, recognition for this work vs new functionality
- Difference in a company: resources, the way teams are organized, definition of what it means to be a team. PETSc also thinks that dividing up gatherings based on topics might be more productive.
- Knowledge within the brain, but not written down. Assumptions built up over time. Curation and maintenance of knowledge.

About 20 prior to the end of the session, around 1:40 pm CDT, try to reach consensus on 3 - 5 high-level challenges your team identified

- Curation and maintenance of knowledge. Compared to maintenance of the code. For the code we've had many conversations, and have tools. Not so for knowledge.
- Onboarding and growth, and evolution of practice and the size and scope of the software.
- Business model for research software sustenance. Applies to all kinds of software, but especially critical where there is a distributed amorphous community involved in its development.
- Recognition, value and career paths for members of the team. And recognition of challenges in managing diverse teams -- who is the gatekeeper, why?