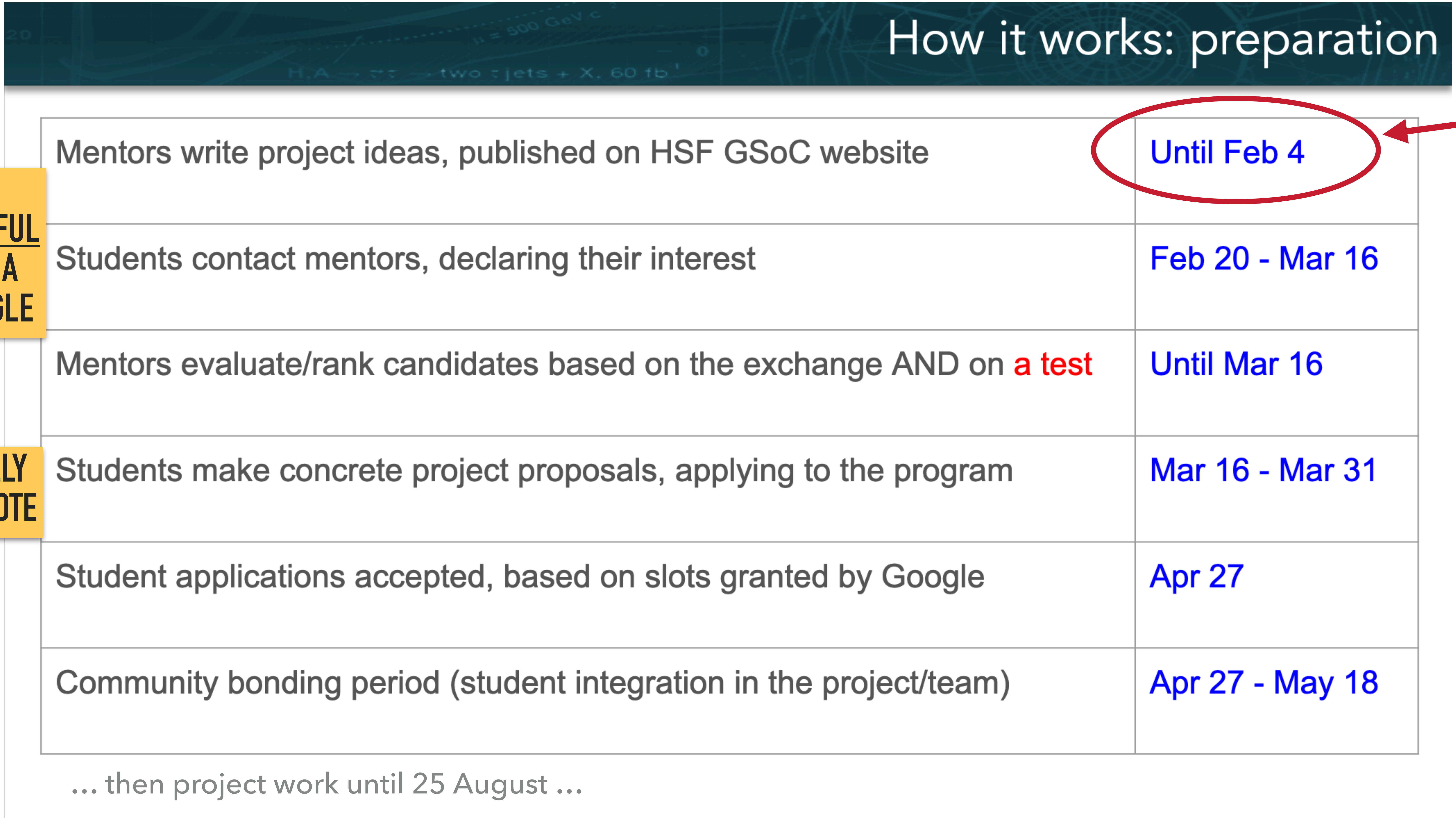
MC GENERATORS & GSOC

STEFAN ROISER

HSF GENERATOR WG MEETING, 24 JAN 2020

GOOGLE SUMMER OF CODE (GSOC) OUTLINE & TIMELINE



How it works: preparation

Mentors write project ideas, published on HSF GSoC website	Until Feb 4
Students contact mentors, declaring their interest	Feb 20 - Mar 16
Mentors evaluate/rank candidates based on the exchange AND on a test	Until Mar 16
Students make concrete project proposals, applying to the program	Mar 16 - Mar 31
Student applications accepted, based on slots granted by Google	Apr 27
Community bonding period (student integration in the project/team)	Apr 27 - May 18

… then project work until 25 August …

VERY SOON !!!!

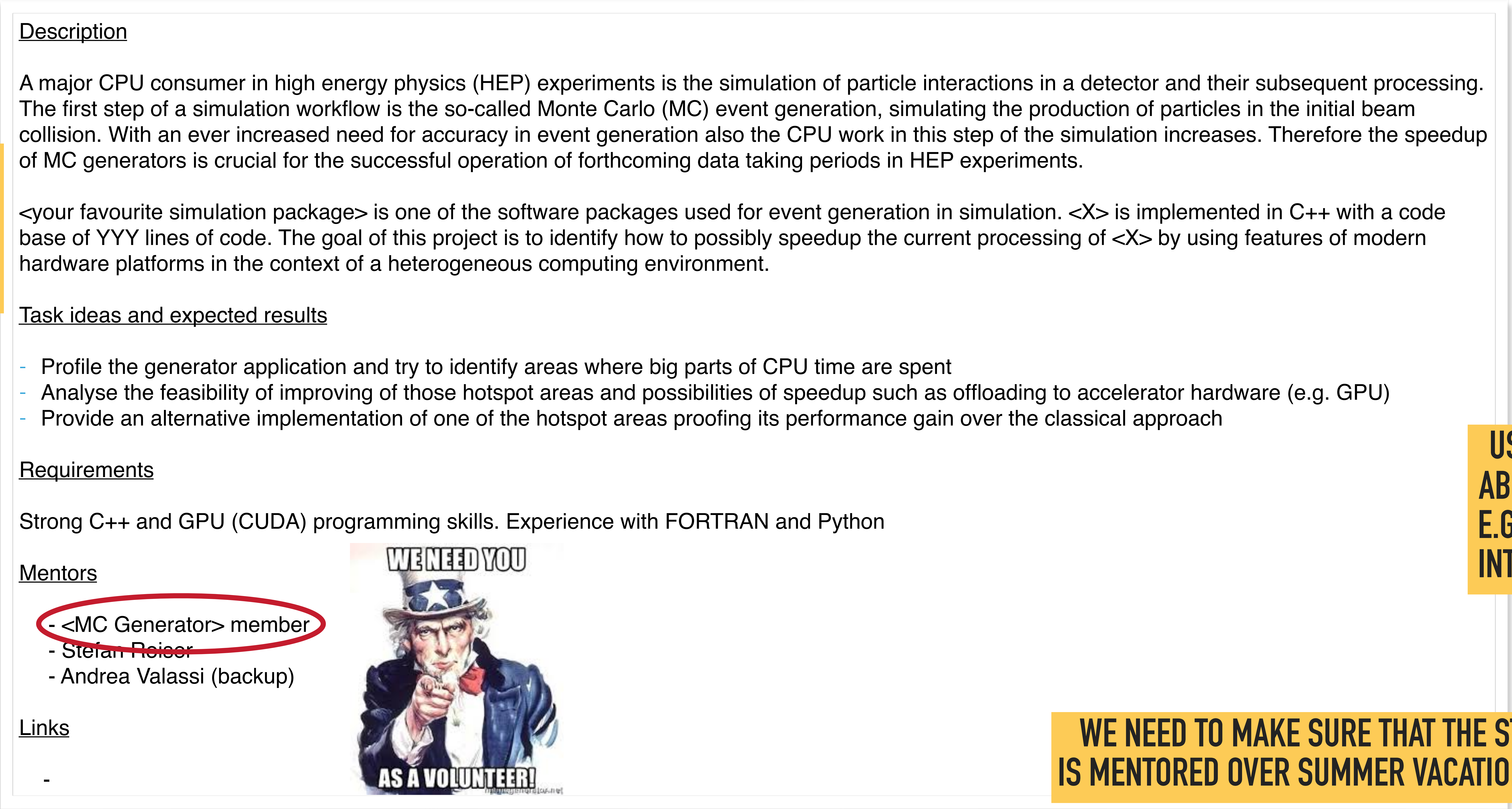
STUDENTS ARE PAID ON SUCCESSFUL COMPLETION OF A PROJECT BY GOOGLE

STUDENTS USUALLY WORK FROM REMOTE

From: https://docs.google.com/presentation/d/1_FIzNoAKLlNMuRsLIMoHhN7EW6_QoUfYaVylwE9pcSQ/edit#slide=id.g6d7bf0887a_0_177

A POSSIBLE GSOC PROJECT

PROFILING E.G. VIA FLAME GRAPHS, INTEL ADVISOR, ETC.



Description

A major CPU consumer in high energy physics (HEP) experiments is the simulation of particle interactions in a detector and their subsequent processing. The first step of a simulation workflow is the so-called Monte Carlo (MC) event generation, simulating the production of particles in the initial beam collision. With an ever increased need for accuracy in event generation also the CPU work in this step of the simulation increases. Therefore the speedup of MC generators is crucial for the successful operation of forthcoming data taking periods in HEP experiments.

<your favourite simulation package> is one of the software packages used for event generation in simulation. <X> is implemented in C++ with a code base of YYY lines of code. The goal of this project is to identify how to possibly speedup the current processing of <X> by using features of modern hardware platforms in the context of a heterogeneous computing environment.

Task ideas and expected results

- Profile the generator application and try to identify areas where big parts of CPU time are spent
- Analyse the feasibility of improving of those hotspot areas and possibilities of speedup such as offloading to accelerator hardware (e.g. GPU)
- Provide an alternative implementation of one of the hotspot areas proofing its performance gain over the classical approach

Requirements

Strong C++ and GPU (CUDA) programming skills. Experience with FORTRAN and Python

Mentors

- <MC Generator> member
- ~~Stefan Roiser~~
- Andrea Valassi (backup)

Links

-

IS THIS TOO MUCH? SHALL WE RESTRICT THE SCOPE?

USE OF HARDWARE ABSTRACTION TOOLS E.G. ALPAKA, KOKOS, INTEL ONEAPI, ETC. ?

WE NEED TO MAKE SURE THAT THE STUDENT IS MENTORED OVER SUMMER VACATION PERIOD