• Answer ANY FOUR questions.
• USE A NEW ANSWER SHEET FOR EACH QUESTION
• Put the question #, your name, person number, and team number on every question sheet that you answer.
• Hand in only the answer sheets. Make sure that you hand in FOUR.

Instructors: Michael Buckley, Carl Alphonce,  & Helene Kershner

The attached exam consists of six essay questions. You must complete any four. Essays are graded on content, explanation, convincing argument, and demonstration of a level of understanding. Free and abstract thought is encouraged, but only in defense of a solid point-of-view. Lists, or bullets, that are essentially correct but with no explanation, are given the minimum allowable credit. Each question is worth 25 points. 20 points will be given for a correct answer. An additional 5 points will be given for a demonstration of a substantial level of understanding (discussion of opposing points of view, for example, or the use of an example from real life, your reading, your team project, or an imagined impact of the subject, on your future job as a software engineer).

Question #1: What does the “dream curve” say about coding?

Question #2: Give and explain 3 reasons (any three) for software project failure.

Question #3: Explain the difference between the problem space and the solution space. Which is reflected in the SS, and which is reflected in the SRS? Who writes, and what is the intended audience for each?

Question #4: Name a software development issue that might have two opposing but valid views. (e.g. designs are sometimes flexible, sometimes follow the spec exactly). Briefly explain the correctness of each view.

Question #5: The 12 XP practices can be characterized as common sense principles which just about any development methodology employs in some manner. Assuming this is essentially correct, explain what distinguishes XP from other methodologies (such as waterfall or the "V" model). Give three or four of the XP principles, and explain how the XP application of these principles, in their interaction and support of each other, results in a methodology which is reactive to change

Question #6: Explain, in general terms, what a refactoring is. Give an example of a refactoring (name the refactoring, and explain what it does). Discuss why tool support for refactoring is desirable.