# CSC 232: Data Structures and Algorithms

**Asn5-1: Software Walkthrough for DEque ADT**

**Due: Friday 10/30/15 before class.**

*Read this short bit of information that will orient you to software walkthrough and code review, then follow the instructions at the end. Our walkthrough will be Friday in class.*

**The Software Walkthrough or Code Review**

Summarized from [Wikipedia - Software Walkthrough](https://en.wikipedia.org/wiki/Software_walkthrough).

In software engineering, a *walkthrough* is a form of software peer review in which a programmer leads a group through the program source code. Participants ask questions and make comments about possible errors, violation of coding standards, and other problems. A walkthrough is typically an open forum and its objective is familiarization.

**Process**

A walkthrough may be quite informal, or may follow the process detailed in IEEE 1028 and outlined in the article on [software reviews](https://en.wikipedia.org/wiki/Software_review).

**Objectives and participants**

A walkthrough is normally organized and directed by the author of the technical document. Any combination of interested or technically qualified personnel (from within or outside the project) may be included as seems appropriate.

**Formal Process: IEEE 1028**

From [Wikipedia - IEEE Formal Software Review](https://en.wikipedia.org/wiki/Software_review%23IEEE_1028_generic_process_for_formal_reviews).

IEEE Standard 1028 defines a common set of activities for "formal" reviews. Differing types of review may apply this structure with varying degrees of rigor, but all activities are mandatory for inspection:

1. **[Entry evaluation]:** The Review Leader uses a standard checklist of entry criteria to ensure that optimum conditions exist for a successful review.
2. **Management preparation:** Responsible management ensure that the review will be appropriately resourced with staff, time, materials, and tools, and will be conducted according to policies, standards, or other relevant criteria.
3. **Planning the review:** The Review Leader identifies or confirms the objectives of the review, organizes a team of Reviewers, and ensures that the team is equipped with all necessary resources for conducting the review.
4. **Overview of review procedures:** The Review Leader, or some other qualified person, ensures (at a meeting if necessary) that all Reviewers understand the review goals, the review procedures, the materials available to them, and the procedures for conducting the review.
5. **[Individual] Preparation:** The Reviewers individually prepare for group examination of the work under review, by examining it carefully for *anomalies* (potential defects), the nature of which will vary with the type of review and its goals.
6. **[Group] Examination:** The Reviewers meet at a planned time to pool the results of their preparation activity and arrive at a consensus regarding the status of the document (or activity) being reviewed.
7. **Rework/follow-up:** The Author of the work product (or other assigned person) undertakes whatever actions are necessary to repair defects or otherwise satisfy the requirements agreed to at the Examination meeting. The Review Leader verifies that all action items are closed.
8. **[Exit evaluation]:** The Review Leader verifies that all activities necessary for successful review have been accomplished, and that all outputs appropriate to the type of review have been finalized.

**Informal Process: Microsoft**

From stackoverflow: [How do you perform code reviews?](http://stackoverflow.com/questions/310813/how-do-you-perform-code-reviews)

Our team tends toward a more informal code review style. New chunks of code, large or complex changes first go through a face to face walk through. Several of us gather in a conference room and the author walks everyone through the changes. The point is to ensure everyone understands why the author did things the way they did.

Before a walk through, participants are expected to have read any specs and other docs and gone through the code on their own. The walk through is more for answering questions and understanding than providing constructive feedback. As the dev manager, I really discourage comments on style in this meeting: our teams have well defined and documented coding standards that folks are expected to follow. We're not draconian about it, but we'll cancel a walk through if myself or one of the leads fells the code isn't ready for a walk through due to style issues.

**Management’s Perspective**

From [www.cio.com](http://www.cio.com) (CIO Magazine) “Running an Effective Code Review”; selected, edited comments.

* *Don’t be afraid to criticize the work under review.*
* **Two Options – pick one depending on the goal**
* *Look at style*
  + A code review should ensure that the software follows accepted guidelines for coding style. A style guide can solve a lot of team problems and prevent silly fights. Christopher Buchino, director of software engineering at GotVMail Communications, says a review is useful at the start of a new project or when someone new joins the development team. “Having uniformity [of style] in the code base is extremely helpful towards maintainability”.
* *Look at everything but style*
  + "It is easy to get distracted by the style of the code," says James Pitts, VP of development and program management at [Embarcadero Technologies](http://www.embarcadero.com/). "The wrong style won't break the product; it is just harder to read."
* Focus on catching items that the compiler and other automated tools miss, says Eric W. Brown, president of Saugus.net. It's far better to find bugs where the code doesn't do exactly what it's supposed to (even if it is clean and appears to work in most cases).
* Schwan, managing partner of Solstice Consulting, points out that code reviews are not meant to be a replacement for unit testing or functional testing — there are more efficient and automated ways for doing that. Nor should it be a formatting exercise. “Some of the worst code reviews I've sat through have involved developers measuring indent spaces or trying to walk through code logic that spans across several modules in their head…”
* Jay Deakins, the founder and president of Deacom, recommends looking for duplication of code, such as multiple versions of a similar function.
* Theron Welch, software mentor at the Microsoft Asia Center for Hardware, suggests that you can calculate the effectiveness of your code reviews by tracking two pieces of information: the average bug fix rate for a team member (which he says usually is one or two per day), and the number of major issues the code review uncovered. Assume you find five major issues in a code review. Your team's major bug fix rate is one per day. If there was no code review, it would have cost approximately five days to fix those bugs. Suppose those issues were fixed in a code review that had five participants. Each person prepared for one hour and the review meeting lasted one hour – total time of 10 hours. Those five major bugs were fixed at 25% of the standard cost **plus** your team gained an additional 30 hours of productive time!

**Requirements**

* Spend approximately 30 minutes to review the code.
* Congratulate yourself on the things that you do understand – you have learned a lot since the beginning of the semester.
* Capture in a Word document five things or questions about the code that you do not understand. Submit in BB \*before\* class on Friday.