Unit 8 cscope test answers Full Version

unit 8 cscope test answers are essential resources for students and programmers preparing for exams or assessments involving cscope, a powerful tool used for browsing and navigating large codebases. Whether you're a beginner or an experienced developer, having accurate and comprehensive answers can significantly improve your understanding of cscope functionalities, improve your test scores, and enhance your ability to efficiently analyze code. In this article, we will delve deep into the key concepts, typical questions, and detailed answers related to Unit 8 cscope test, ensuring you are well-equipped to excel in your evaluations.

Understanding cscope and Its Importance

Before exploring the test answers, it's vital to comprehend what cscope is and why it's a critical tool for developers.

What is cscope?

cscope is an interactive, text-based tool designed for browsing source code in C, C++, and other programming languages. It allows developers to:

- Search for functions, definitions, and variables.
- Find all references to a particular symbol.
- Navigate between different parts of code efficiently.
- Understand complex code structures in large projects.

The Significance of cscope in Software Development

Using cscope improves productivity by:

- Reducing time spent on navigating code.
- Enhancing understanding of code dependencies.
- Facilitating debugging and code reviews.
- Supporting large-scale code analysis.

Overview of Unit 8 cscope Test

The Unit 8 cscope test often covers various aspects, including configuration, command usage, and

practical application scenarios. Typical questions evaluate your comprehension of cscope commands, database creation, integration with editors like Vim, and interpretation of output results.

Common Topics Covered in the Test

- Setting up cscope databases.
- Using cscope commands for searching symbols, functions, and files.
- Integrating cscope with IDEs or editors.
- Navigating code using cscope.
- Troubleshooting common issues.

Key Questions and Accurate Answers for Unit 8 cscope Test

Below are some of the most frequently asked questions in the test, along with detailed, accurate answers to help you prepare thoroughly.

1. How do you create a cscope database?

- Navigate to the directory containing your source code.
- Use the command:cscope -b -q -k

where:

- -b: build the cross-reference only.
- -q: generate an inverted index for quick searching.
- -k: use kernel-style source code (excluding /usr/include).

- Ensure your source files are listed in a cscope.files file or specify them directly.
- After running the command, a cscope.out file will be created, which is the database used for searches.

2. How can cscope be integrated with Vim?

To integrate cscope with Vim:

- Generate the cscope database as described above.
- In Vim, run::cscope add cscope.out

to add the database to your current session.

- Use cscope commands within Vim:

- **:cs find text** ? Find all functions, variables, etc., matching .
- **:cstag** ? Jump to the definition of .

- Bind cscope commands to shortcuts for faster navigation (optional).

3. What are the main cscope commands and their functions?

- **Find this symbol**: Search for all references to a specific symbol.
- **Find global definition**: Locate the definition of a function or variable.
- **Find functions called by**: List all functions called by a specified function.
- **Find functions calling**: List all functions that call a specific function.
- **Find files** : Search for files containing a specific pattern or symbol.
- **Find text string**: Search for a specific string in the codebase.

4. How do you troubleshoot common issues with cscope?

Some frequent problems include:

- **Database not updating:** Rebuild the database using the *cscope -b* command.
- **Incorrect source files listed:** Ensure all relevant source files are included in the *cscope.files* list.
- **Integration problems with editors:** Verify the correct path to *cscope.out* and proper configuration in your editor.
- **Commands returning no results:** Confirm the symbol exists in your code and that your database is current.

Best Practices for Using cscope Effectively

To maximize the benefits of cscope, consider the following best practices:

1. Keep Your Database Up-to-Date

- Regularly rebuild the cscope database, especially after large code changes.
- Automate database updates in your build or version control process.

2. Use Multiple Search Options

- Combine cscope with other tools like ctags for comprehensive code navigation.
- Use the -q option for faster searches in large codebases.

3. Organize Source Files Properly

- Maintain a clear and updated list of source files.
- Use a dedicated file like cscope.files for easy management.

4. Integrate with Your Development Environment

- Configure your IDE or editor for seamless navigation.
- Customize keybindings for quick access to cscope features.

Advantages of Using cscope for Code Navigation

Implementing cscope offers numerous advantages:

- Accelerates the process of locating functions, variables, and definitions.
- Facilitates understanding of code flow and dependencies.
- Supports large projects with complex code structures.
- Enhances debugging and troubleshooting efficiency.
- Integrates well with popular editors like Vim, Emacs, and others.

Conclusion

Mastering the unit 8 cscope test answers is crucial for anyone involved in software development, especially those working extensively with large C/C++ codebases. Understanding how to create, manage, and utilize cscope databases, along with integrating cscope into your development workflow, can drastically improve your productivity and code comprehension. Remember to regularly practice with typical test questions, familiarize yourself with cscope commands, and follow best practices for database maintenance. With dedication and the right knowledge, you'll be well-prepared to excel in your cscope assessments and leverage the tool to write better, more organized code.

Additional Resources for Learning cscope

- Official cscope documentation and man pages.

- Tutorials on integrating cscope with Vim or Emacs.
- Community forums and coding blogs discussing advanced cscope features.
- Open-source projects using cscope for large-scale code analysis.

By leveraging these resources, you can deepen your understanding and become proficient in using cscope for all your development needs.

Unit 8 cscope test answers have become an essential resource for students and professionals preparing for programming and software development assessments. Cscope, a powerful tool for browsing C source code, is widely used in many academic and professional settings to facilitate code navigation, understanding, and debugging. As a result, mastering its features and functionalities through test questions and answers can significantly enhance one's ability to work efficiently with large codebases. This comprehensive review aims to explore the importance of Unit 8 cscope test answers, their structure, key topics covered, and how they serve as an effective learning aid.

Understanding Cscope and Its Relevance in Programming

What is Cscope?

Cscope is an interactive program that allows developers to browse through C source code efficiently. It provides a text-based interface to search for functions, variables, macros, and other code elements, making it easier to understand complex codebases without manually scrolling through files.

Features of Cscope:

- Fast and efficient code navigation
- Search for symbol definitions, function calls, and references
- Cross-reference list generation
- Supports multiple views and queries simultaneously
- Integration with editors like Vim and Emacs

Pros:

- Handles large codebases gracefully
- Simplifies understanding of unfamiliar code
- Enhances debugging efficiency

Cons:

- Requires familiarity with command-line interfaces
- Limited to C and C++ languages
- May have a steep learning curve for beginners

Given its significance, many educational institutions include cscope in their curriculum, often culminating in tests or assignments?hence the importance of accurate unit 8 cscope test answers.

Overview of Unit 8 Cscope Test Content

Unit 8 typically covers advanced usage and features of cscope, including:

- Setting up cscope databases
- Performing complex searches
- Integrating cscope with development environments
- Customizing cscope configurations
- Practical exercises involving code navigation

The test questions aim to evaluate students? understanding of these concepts, their ability to apply commands effectively, and their problem-solving skills related to code analysis.

Key Topics Covered in Unit 8 Cscope Test Answers

1. Setting Up and Using Cscope

This section tests knowledge about initializing cscope databases, creating cross-reference files, and launching cscope for interactive queries.

- Creating the database: Using commands like `cscope -b -q -k` to build the database.
- Updating the database: Rebuilding when source files change.
- Launching cscope: Running `cscope -d` to open the interactive mode.

Sample Question:

How do you generate a cscope database for a directory of source files?

Answer:

Use the command `cscope -b -q -k` inside the source directory, which creates the `cscope.out` database and auxiliary files.

Pros:

- Automates code indexing
- Ensures quick searches in large projects

Cons:

- Requires manual database updates after code changes

2. Navigating Code with Cscope

Test questions often focus on executing searches such as:

- Finding all functions called by a specific function
- Locating all references to a variable
- Identifying where a function or variable is defined

Common Commands:

| Command | Description |

| --- | --- |

| `s` | Search for a symbol (function, variable) |

| `g` | Find global definitions |

| `t` | Find all references to a symbol |

| `c` | Find functions calling a specific function |

Sample Question:

Which cscope command allows you to find all functions that call a particular function?

Answer:

Enter `c` in cscope and specify the function name.

Features:

- Rapidly trace code flow
- Facilitates debugging and code comprehension

Cons:

- Requires familiarity with command types
- Might be overwhelming for first-time users

3. Integrating Cscope with Development Environments

This segment emphasizes how to embed cscope into editors like Vim or Emacs for seamless code navigation.

Integration Techniques:

- Using Vim's `:cscope` command
- Configuring `.vimrc` for automatic cscope database loading
- Using plugins for enhanced functionality

Sample Question:

How can you integrate cscope with Vim for efficient source code browsing?

Answer:

By generating a cscope database and configuring Vim with commands like `:cscope add cscope.out`, enabling quick searches within the editor.

Features:

- Streamlines workflow

- Reduces context switching
- Improves productivity

Cons:

- Requires initial configuration
- Compatibility issues may arise with certain editors

Strengths and Limitations of Using Cscope and Test Answers

Strengths

- Enhanced Learning: Test answers provide step-by-step solutions, helping students grasp complex commands and workflows.
- Time-Saving: Well-prepared answers enable quick revision and practice.
- Practical Focus: Emphasize real-world applications, preparing students for actual coding scenarios.
- Coverage of Common Errors: Highlight typical mistakes and how to avoid them.

Limitations

- Dependence on Memorization: Over-reliance on answers might hinder deep understanding.
- Context Specificity: Answers tailored to specific questions might not translate well to different problems.
- Potential for Outdated Information: As tools evolve, some answers may become obsolete if not updated regularly.

How to Use Unit 8 Cscope Test Answers Effectively

- Understand, Don't Memorize: Use answers as guides to understand the reasoning behind commands and procedures.
- Practice Regularly: Apply answers in actual coding projects to reinforce learning.
- Update Knowledge: Stay current with the latest cscope features and best practices.
- Combine with Hands-on Exercises: Use answers alongside practical tasks to solidify skills.

Conclusion

Unit 8 cscope test answers serve as a vital educational resource for mastering code navigation and analysis tools in C programming. They offer comprehensive insights into setting up cscope, executing complex searches, integrating with development environments, and troubleshooting common issues. While they significantly aid in learning, users should aim to develop a deep understanding of underlying concepts rather than solely relying on memorized answers. Combining these resources with hands-on practice will prepare students and professionals to utilize cscope effectively, ultimately leading to more efficient and error-free software development processes. As the landscape of development tools continues to evolve, staying updated and adaptable remains key to leveraging cscope?s full potential.

QuestionAnswer What is the purpose of Unit 8 CSCOPE test answers? Unit 8 CSCOPE test answers are designed to help students review and prepare for their assessments by providing correct responses to questions related to the unit's curriculum. Are the Unit 8 CSCOPE test answers available for free online? Yes, many educational websites and student forums offer free access to Unit 8 CSCOPE test answers, but students should use them ethically and as a study aid rather than for cheating. How can I effectively use Unit 8 CSCOPE test answers to improve my understanding? Use the answers to check your responses after attempting the questions yourself, review explanations for any mistakes, and revisit the related lessons to strengthen your grasp of the concepts. What topics are typically covered in Unit 8 CSCOPE tests? Unit 8 CSCOPE tests often cover topics like algebraic expressions, functions, geometry, data analysis, and other key concepts from the curriculum relevant to that unit. Is relying solely on Unit 8 CSCOPE test answers recommended for exam success? No, relying solely on test answers is not recommended; instead, use them as a supplementary resource alongside thorough studying, practice, and understanding of the material for true success.

Related keywords: cscope, unit 8 test, programming questions, coding answers, exam solutions, software development, code debugging, programming test answers, cscope tutorial, test preparation

Microsoft Test Manager Tutorial

Microsoft Test Manager Tutorial: A Comprehensive Guide to Efficient Test Management

In the world of software development, ensuring the quality and reliability of your applications is paramount. Microsoft Test Manager (MTM) is a powerful tool designed to streamline the testing process, enabling teams to plan, execute, and track testing efforts with ease. Whether you're a beginner or looking to enhance your testing workflows, this **Microsoft Test Manager tutorial** will provide you with a step-by-step guide to harnessing MTM's full potential for your projects.

Understanding Microsoft Test Manager

Before diving into the tutorial, it's essential to grasp what Microsoft Test Manager is and how it integrates into the broader Visual Studio ecosystem.

What is Microsoft Test Manager?

Microsoft Test Manager is a test management tool integrated with Visual Studio Team Services (VSTS) and Azure DevOps. It provides an intuitive interface for managing test plans, creating and executing test cases, and tracking testing progress. MTM supports both manual and exploratory testing, making it versatile for various testing scenarios.

Key Features of Microsoft Test Manager

- Test Planning: Create and organize test plans, test suites, and test cases.
- Test Execution: Execute manual tests and record results efficiently.
- Test Tracking: Monitor testing progress and generate detailed reports.
- Exploratory Testing: Conduct ad-hoc testing sessions with session-based testing capabilities.
- Integration: Seamlessly connects with Azure DevOps for continuous testing workflows.

Setting Up Microsoft Test Manager

Getting started with MTM involves installing and configuring the tool for your environment.

Prerequisites

- Visual Studio Enterprise or Professional edition (with test management features).
- Azure DevOps account or TFS (Team Foundation Server) setup.
- Appropriate permissions to access test management features.

Installing Microsoft Test Manager

- Download the Visual Studio Installer from the official Microsoft website.
- Choose the edition that includes testing tools (e.g., Visual Studio Enterprise).
- During installation, select the "Testing Tools" workload to install MTM.
- Complete the installation and launch Visual Studio.
- Connect to your Azure DevOps project or TFS server within Visual Studio.

Creating and Managing Test Plans

A well-structured test plan is the foundation of successful testing. MTM simplifies organizing your testing efforts through test plans and suites.

Creating a Test Plan

- Open Microsoft Test Manager from Visual Studio or the Azure DevOps portal.
- Navigate to the "Test Plans" section.
- Click "New Test Plan" and provide a descriptive name, description, and start/end dates.
- Associate the test plan with a specific build or release if needed.
- Save the test plan to begin adding test cases.

Organizing Test Suites

Test suites help categorize test cases based on features, modules, or testing strategies.

- **Static Test Suites:** Manually organize test cases into logical groups.
- **Requirement-Based Test Suites:** Link test cases to specific requirements or user stories.
- **Query-Based Test Suites:** Dynamically generate test cases based on queries.

Adding Test Cases

- Within your test suite, click "New Test Case" or import existing cases.
- Define the test case steps, expected results, and associated requirements.
- Assign priority, owner, and other metadata to facilitate management.
- Save the test case to include it in your suite.

Executing Tests with Microsoft Test Manager

Once your test cases are organized, the next step is executing tests and recording results.

Manual Test Execution

- Open the test plan and select the test suite to execute.
- Click "Run" on the test case you wish to execute.
- Follow the test steps as documented, marking each step as "Passed" or "Failed."

- If a test fails, record detailed bug information directly from the test runner.
- Complete the test run, and the results will be saved automatically.

Using Action Recording and Data Collection

MTM allows capturing detailed logs and screenshots during test execution, aiding in defect analysis and reproducibility.

Exploratory Testing

For ad-hoc testing, MTM offers session-based testing:

- Schedule a testing session with session details.
- Conduct testing while documenting observations and findings.
- Record bugs and issues discovered during the session.

Tracking and Reporting Testing Progress

Effective test management requires insight into testing status and quality metrics.

Monitoring Test Results

- Navigate to the "Test Results" section in MTM or Azure DevOps.
- Review individual test case outcomes and overall progress.
- Filter results by tester, status, or test plan for detailed analysis.

Generating Reports

MTM provides various built-in reports to visualize testing status:

- Test Results Summary
- Test Case Progress and Trends
- Bugs and Defects Reports

Integrating with Bug Tracking

During testing, bugs can be logged directly from MTM, linked to specific failed test cases, ensuring traceability and quick resolution.

Advanced Tips and Best Practices

Maximize your testing efficiency with these expert tips:

Automate Repetitive Tests

Integrate MTM with automated testing frameworks like MSTest, NUnit, or Selenium to run automated tests alongside manual efforts.

Maintain Test Cases Regularly

Keep your test cases updated with application changes to ensure relevance and accuracy.

Use Tags and Filters

Leverage tags and filters to quickly locate and execute specific test cases based on criteria like priority, feature, or owner.

Collaborate Effectively

Encourage team collaboration by sharing test plans and results, and conducting regular review meetings to discuss testing progress.

Conclusion

Microsoft Test Manager is an essential tool for teams aiming to implement structured and efficient testing workflows. From creating comprehensive test plans and organizing test suites to executing tests and tracking outcomes, MTM provides a centralized platform that enhances test visibility and quality assurance. By following this **Microsoft Test Manager tutorial**, you can streamline your testing process, improve defect detection, and deliver higher-quality software products. Remember to stay updated with the latest features and best practices to maximize your testing success with MTM.

Microsoft Test Manager tutorial: A Comprehensive Guide to Streamlining Your Testing Process

Microsoft Test Manager (MTM) is a powerful tool integrated within the Azure DevOps suite, designed to facilitate efficient test planning, management, and execution. Whether you are a QA professional, a developer, or a project manager, mastering MTM can significantly enhance your testing workflows, improve defect detection rates, and ensure higher quality software releases. This tutorial aims to provide an in-depth overview of Microsoft Test Manager, guiding you through its features, setup, best practices, and real-world applications.

Introduction to Microsoft Test Manager

Microsoft Test Manager is a dedicated testing management environment that helps teams plan, execute, and track testing activities seamlessly. Originally part of Visual Studio Test Professional, MTM is now integrated within Azure DevOps, offering a unified platform for Agile testing, exploratory testing, and manual test case management.

Key Features of Microsoft Test Manager

- Test Planning: Create and organize test plans, test suites, and test cases.
- Test Execution: Run manual tests, record results, and capture screenshots.
- Test Management: Track defects, manage test configurations, and generate reports.
- Integration: Seamlessly connect with Azure DevOps for continuous integration and automated testing workflows.
- Exploratory Testing: Record exploratory sessions with detailed notes and recordings.

Setting Up Microsoft Test Manager

Before diving into testing, proper setup of MTM is essential.

Prerequisites

- A valid Azure DevOps account with appropriate permissions.
- Installed Visual Studio Test Professional or access via Azure DevOps.
- Access to a test environment or lab machines.

Installation and Configuration

- Download and Install: Download Microsoft Test Manager from the Visual Studio downloads page or via Azure DevOps.
- Connect to Azure DevOps: Launch MTM and connect it to your Azure DevOps project by signing in with your credentials.
- Configure Test Environments: Set up test machines, configurations, and network environments to align with your testing needs.

Tips for Effective Setup

- Organize your test plans hierarchically for easier management.
- Maintain a clear mapping between test environments and real-world configurations.
- Regularly update test artifacts to ensure they reflect current application versions.

Creating and Managing Test Plans

Test planning is the foundation of effective testing.

Creating a Test Plan

- In MTM, navigate to the "Test Plans" section.
- Click "New Test Plan" and provide a descriptive name, start/end dates, and any relevant details.
- Assign ownership and stakeholders.

Organizing Test Suites

- Static Test Suites: Manually added test cases grouped logically.
- Requirement-based Suites: Linked to user stories or requirements for traceability.
- Query-based Suites: Dynamic grouping based on queries.

Best Practices

- Break down large test plans into smaller, manageable suites.
- Link test cases to user stories or bugs for better traceability.
- Regularly review and update test plans to reflect current project scope.

Designing and Managing Test Cases

Creating effective test cases is critical to uncovering defects early.

Writing Test Cases

- Clear, concise steps with expected results.
- Use descriptive titles and tags for easy filtering.
- Include preconditions and postconditions.

Reusing Test Cases

- Modularize test steps for reuse across different test cases.
- Attach relevant documentation, screenshots, or scripts.

Organizing Test Cases

- Group related test cases into shared test suites.
- Use labels and tags for categorization.

Tips for Effective Test Cases

- Prioritize test cases based on risk and impact.
- Keep test cases up-to-date with application changes.
- Incorporate exploratory testing notes within test cases or as separate sessions.

Test Execution and Result Recording

Executing tests efficiently and accurately recording results is vital.

Running Manual Tests

- Select a test case from your test suite.
- Follow each step, observing the actual behavior.
- Record outcomes: Passed, Failed, Blocked, or Not Applicable.
- Capture screenshots or record videos if needed.

Recording Defects

- Link defects directly to failed test cases.
- Provide detailed descriptions, steps to reproduce, and severity.
- Attach logs or screenshots to aid debugging.

Managing Test Runs

- Use test configurations to run multiple test cases under different environments.
- Schedule and assign test runs to team members.
- Monitor real-time progress and results.

Pros and Cons of Test Execution Features

Pros:

- Streamlined process for manual testing.
- Rich media capture for detailed documentation.
- Easy defect linkage and traceability.

Cons:

- Manual process can be time-consuming for large test suites.
- Limited automation capabilities within MTM itself.

Integrating Automated Testing with Microsoft Test Manager

While MTM excels at manual testing, integration with automated testing frameworks enhances overall efficiency.

Integration with Visual Studio Test Automation

- Use Visual Studio Test tools to develop automated test scripts.
- Link automated tests to test cases in MTM.
- Run automated tests as part of continuous integration pipelines.

Benefits of Integration

- Reduces manual effort.
- Ensures consistency between manual and automated testing.
- Facilitates comprehensive test coverage.

Limitations

- Setup can be complex for beginners.
- Requires scripting knowledge for automation.

Reporting and Tracking

Effective reporting provides insights into testing progress and quality metrics.

Built-in Reports

- Test Run Summary
- Test Results Trend
- Defect Status and Age
- Requirements Traceability Matrix

Custom Reports

- Use Azure DevOps Analytics or Power BI for tailored dashboards.
- Export data for external analysis.

Best Practices

- Regularly review reports to identify bottlenecks.
- Use metrics to prioritize testing efforts.
- Maintain up-to-date dashboards for stakeholder visibility.

Best Practices for Using Microsoft Test Manager

- Plan Early: Incorporate testing early in the development lifecycle.
- Maintain Test Artifacts: Keep test cases and plans updated with application changes.
- Leverage Linkages: Connect requirements, test cases, defects, and build versions.
- Automate Where Possible: Use automation to handle repetitive and regression testing.
- Collaborate: Engage team members through shared test plans and results.
- Review Regularly: Conduct test reviews and retrospectives for continuous improvement.

Common Challenges and How to Overcome Them

Challenge: Managing Large Test Suites

Solution: Break down into smaller, manageable suites; use query-based suites for dynamic grouping.

Challenge: Keeping Test Cases Up-to-date

Solution: Establish a review cycle aligned with development sprints; assign ownership.

Challenge: Integration Complexity

Solution: Invest in automation and continuous integration pipelines; use documentation and training.

Challenge: Limited Automation within MTM

Solution: Use external automation tools integrated with Azure DevOps, such as Selenium or Coded UI.

Conclusion

Microsoft Test Manager is a robust tool that, when used effectively, can significantly boost your testing efficiency, coverage, and traceability. Its comprehensive features for test planning, execution, defect management, and reporting make it indispensable for teams adopting Agile and DevOps practices. While there are challenges, especially around automation and managing large test repositories, with proper setup, training, and best practices, MTM can become a cornerstone of your software quality assurance process.

By following this tutorial, you should now have a clear understanding of how to leverage Microsoft Test Manager to streamline your testing operations, improve collaboration, and ultimately deliver higher quality software to your users. Remember, continuous learning and adaptation are key to maximizing the benefits of MTM in your projects.

QuestionAnswer What are the main features of Microsoft Test Manager (MTM)? Microsoft Test Manager (MTM) provides features such as test planning, manual test execution, test case management, bug tracking integration, and collaboration tools to streamline the testing process within Visual Studio and Azure DevOps. How do I create and organize test plans in Microsoft Test Manager? In MTM, you can create test plans by navigating to the 'Test Plans' hub, clicking 'New Test Plan,' and defining its name and details. You can then add test suites and test cases to organize tests based on requirements, features, or test types for better management. Can I automate tests using Microsoft Test Manager, and how is it integrated with other tools? While MTM primarily supports manual testing, it integrates seamlessly with Visual Studio and Azure DevOps, enabling automation through test scripts and frameworks like MSTest, Selenium, or CodedUI. Automated tests can be linked to test cases for comprehensive test management. What are the best practices for using Microsoft Test Manager effectively? Best practices include maintaining detailed and reusable test cases, organizing tests into logical suites, utilizing shared steps for common actions, integrating with bug tracking, and regularly updating test plans based on project changes to ensure thorough testing coverage. Is Microsoft Test Manager suitable for Agile development environments? Yes, MTM is well-suited for Agile teams as it supports iterative testing, easy

integration with Azure DevOps Agile tools, and flexible test planning, allowing teams to adapt testing activities to sprint cycles and continuous delivery workflows.

Related keywords: Microsoft Test Manager, TFS testing, test planning, test case management, automated testing, test execution, bug tracking, test lab management, test reporting, test automation tools

Read the full article online: Microsoft Test Manager Tutorial