Congratulations! You passed!
TO PASS 80% or higher

Keep Learning

GRADE
92.85%

Module 4 Graded Quiz

LATEST SUBMISSION GRADE
92.85%

1. These three areas are core tenants of a bug management and mitigation strategy. **1 / 1 point**

- [x] Eliminate

Correct
Correct, each of these three areas are core tenants of a bug management and mitigation strategy:

- Detection
- Diagnose
- Eliminate

- [x] Diagnose

Correct
Correct, each of these three areas are core tenants of a bug management and mitigation strategy:

- Detection
- Diagnose
- Eliminate

- [x] Detect

Correct
Correct, each of these three areas are core tenants of a bug management and mitigation strategy:

- Detection
- Diagnose
- Eliminate

- [ ] Migrate
- [ ] Evaluate

2. A bug/defect lifecycle should be integrated and part of this? **1 / 1 point**

- (•) Software development lifecycle (SDLC)
- ( ) Cross industry standard procedure for data management (CRISP-DM)
- ( ) National standard for mainframe management (NSMM)

Correct
Correct, a bug/defect life cycle should be included as part of a larger software development life cycle.

3. Which of the following apply to a bug/defect life cycle (select all that apply)? **1 / 1 point**

- [x] Tracks different states of bug.

Correct
Correct, a bug/defect lifecycle starts as soon as any new defect is found by a tester and comes to an end when a tester closes that defect assuring that it won't get reproduced again.

- [x] Comes to an end when a tester closes that defect.

Correct
Correct, a bug/defect lifecycle starts as soon as any new defect is found by a tester and comes to an end when a tester closes that defect assuring that it won't get reproduced again.

- [x] Starts as soon as any new defect is found by a tester.

Correct
Correct, a bug/defect lifecycle starts as soon as any new defect is found by a tester and comes to an end when a tester closes that defect assuring that it won't get reproduced again.

- [ ] Leaves open the possibility that a resolved bug may get reproduced again.

4. This is an extremely critical part of the bug/defect lifecycle that should not be ignored and helps avoid any confusion about the bug, its status, etc., **1 / 1 point**

- ( ) Collaboration
- (•) Documentation
- ( ) Conversation

Correct
Correct, documentation is an extremely critical part of the bug/defect lifecycle that should not be ignored and helps avoid any confusion about the bug, its status, etc.,

5. Which of the following are items on a bug/defect checklist (select all that apply)? **1 / 1 point**

- [x] Set breakpoints.

Correct
Correct, each of the following are items on a bug/defect checklist:

- Set monitoring for the program.
- Set breakpoints.
- Set other monitoring options as needed.
- Initiate the program.
- End the test session.

- [x] Initiate the program.

Correct
Correct, each of the following are items on a bug/defect checklist:

- Set monitoring for the program.
- Set breakpoints.
- Set other monitoring options as needed.
- Initiate the program.
- End the test session.

- [x] Set monitoring for the program.

Correct
Correct, each of the following are items on a bug/defect checklist:

- Set monitoring for the program.
- Set breakpoints.
- Set other monitoring options as needed.
- Initiate the program.
- End the test session.

- [x] Set other monitoring options as needed.

Correct
Correct, each of the following are items on a bug/defect checklist:

- Set monitoring for the program.
- Set breakpoints.
- Set other monitoring options as needed.
- Initiate the program.
- End the test session.

6. These are the two primary COBOL debugging options available: **1 / 1 point**

- [x] Interactive debugging

Correct
Correct, the two different debugging approaches are: source-language debugging and interactive debugging.

- [ ] Intermittent debugging
- [x] Source language debugging

Correct
Correct, the two different debugging approaches are: source-language debugging and interactive debugging.

- [ ] Event driven debugging

7. This type of debugging tests for language elements, compiler options, and listing outputs that make debugging easier. **1 / 1 point**

- ( ) COBOL-like commands debugging
- (•) Source language debugging
- ( ) Interactive debugging

Correct
Correct, source language debugging tests for language elements, compiler options, and listing outputs that make debugging easier.

8. These commands support coding actions to be taken at breakpoints and provided in a syntax similar to the COBOL programming language. **0 / 1 point**

COBOL

Incorrect
Incorrect, revisit the COBOL Debugging Options lesson to learn COBOL-like commands support high level coding actions to be taken at breakpoints and provided in a syntax similar to the COBOL programming language.

9. Which of the following should be part of small test cases used in debugging to simulate the larger application (select all that apply)? **1 / 1 point**

- [x] Errors in program logic

Correct
Correct, the following should be part of small test cases used in debugging to simulate the larger application:

- Errors in program logic
- Input-output errors
- Mismatches of data types
- Uninitialized data

- [x] Input-output errors

Correct
Correct, the following should be part of small test cases used in debugging to simulate the larger application:

- Errors in program logic
- Input-output errors
- Mismatches of data types
- Uninitialized data

- [x] Mismatches of data types

Correct
Correct, the following should be part of small test cases used in debugging to simulate the larger application:

- Errors in program logic
- Input-output errors
- Mismatches of data types
- Uninitialized data

- [x] Uninitialized data

✓ **Correct**

Correct, the following should be part of small test cases used in debugging to simulate the larger application:

- Errors in program logic
- Input-output errors
- Mismatches of data types
- Uninitialized data

10. This tool offers debugging and code coverage for z/OS applications written in COBOL, PL/I, C/C++ and Assembler. **1 / 1 point**

- (●) IBM Z/OS DEBUGGER
- ( ) IBM Db2
- ( ) IBM Z Open Editor

✓ **Correct**

Correct, IBM z/OS Debugger offers debugging and code coverage for z/OS applications written in COBOL, PL/I, C/C++ and Assembler.

11. The IBM z/OS Debugger and provides a 3270 user interface and remote debugging through Eclipse. **1 / 1 point**

- (●) True
- ( ) False

✓ **Correct**

Correct, the IBM z/OS Debugger and provides a 3270 user interface and remote debugging through Eclipse.

12. This is a type of configuration that allows a developer to debug a program under certain conditions. **1 / 1 point**

- ( ) CICD persona
- (●) De-bugger profiles
- ( ) Port overrides

✓ **Correct**

Correct, de-bugger profiles is a type of configuration that allows a developer to debug a program under certain conditions.

13. When using the IBM z/OS Debugger the application must not include any other languages. **1 / 1 point**

- (●) True
- ( ) False

✓ **Correct**

Incorrect, revisit the IBM z/OS Debugger lesson to learn when using the IBM z/OS Debugger the application may include other languages.

14. Which of the following modes may the IBM z/OS Debugger be run (select all that apply). **1 / 1 point**

- [x] Batch mode.

✓ **Correct**

Correct, the IBM z/OS Debugger can be used to debug programs in batch mode, interactively in full-screen mode, or in remote debug mode.

- [x] Interactively in full-screen mode.

✓ **Correct**

Correct, the IBM z/OS Debugger can be used to debug programs in batch mode, interactively in full-screen mode, or in remote debug mode.

- [x] Remote debug mode

✓ **Correct**

Correct, the IBM z/OS Debugger can be used to debug programs in batch mode, interactively in full-screen mode, or in remote debug mode.

- [ ] None of the above.