Study Guide Questions with Answers - Part 1 of 2

Recursion

1. Which of the following best describes recursion in programming?

- a) A function that calls itself.
- b) A function that calls another function.
- c) A loop that iterates over a collection.
- d) A function that runs indefinitely.

Answer: a

Analysis:

- **Option a)** is correct because recursion is defined as a function that calls itself.
- **Option b)** is incorrect because it describes a general function call, not recursion.
- **Option c)** is incorrect because it describes iteration, not recursion.
- **Option d)** is incorrect because recursion does not inherently mean indefinite execution.

2. In a recursive function, what is the purpose of the condition that checks if the input has reached a certain value?

- a) To create an infinite loop.
- b) To stop the recursion and return a result.
- c) To call another function.
- d) To print a message to the console.

Answer: b

Analysis:

- **Option a)** is incorrect because the stopping condition is meant to prevent infinite loops.
- **Option b)** is correct because the stopping condition (base case) terminates the recursion and returns a result.
- **Option c)** is incorrect because the stopping condition does not call another function.
- **Option d)** is incorrect because the purpose is to stop recursion, not to print messages.

3. Given the following code, how many times is the `printHello` function called when `printHello(3)` is executed?

```
void printHello(int n)
{
    if (n <= 0) return;
    Console.WriteLine("Hello");
    printHello(n - 1);
}
```

- a) 1
- b) 2
- c) 3
- d) 4

Answer: c

Analysis:

- The function `printHello` calls itself recursively, decrementing `n` by 1 each time until `n` becomes 0.
- **Option a)** is incorrect because the function is called more than once.
- **Option b)** is incorrect because the function is called more than twice.
- **Option c)** is correct because `printHello(3)` results in the function being called with `n = 3, 2, 1`, for a total of 3 calls.
- **Option d)** is incorrect because the function stops calling itself when `n` reaches 0, resulting in only 3 calls.

4. What will be the output of the following recursive function call `Sum(4)`?

```
int Sum(int n)
{
    if (n == 0) return 0;
    return n + Sum(n - 1);
}
```

- a) 4
- b) 10
- c) 15
- d) 20

Answer: b

Analysis:

- The `Sum` function calculates the sum of all integers from `n` down to 0.
- **Option a)** is incorrect because the sum is more than just 4.
- **Option b)** is correct because `Sum(4)` results in `4 + 3 + 2 + 1 + 0 = 10`.
- **Option c)** is incorrect because the sum 15 would correspond to a different series.

- **Option d)** is incorrect because the sum 20 is too high for the given input.

5. What is a common issue that can arise in recursive functions if the stopping condition is not properly handled?

- a) Infinite loops
- b) Compilation errors
- c) Stack overflow
- d) Syntax errors

Answer: c

Analysis:

- **Option a)** is incorrect because while infinite loops are a concern, the specific issue in recursion is stack-related.
- **Option b)** is incorrect because the stopping condition itself does not affect compilation.
- **Option c)** is correct because failing to handle the stopping condition properly can lead to stack overflow due to excessive recursive calls.
- **Option d)** is incorrect because syntax errors are related to code structure, not recursion logic.

Practical Problem Solving

1. Which of the following activities is part of the design phase in the SDLC?

- a) Writing code for the application
- b) Defining user requirements
- c) Creating architectural diagrams
- d) Testing the application

Answer: c

Analysis:

- **Option a)** is part of the development phase.
- **Option b)** is part of the requirements analysis phase.
- **Option c)** is correct because creating architectural diagrams is a key activity in the design phase.
- **Option d)** is part of the testing phase.

2. What does decomposition in algorithmic thinking help achieve?

- a) Combining multiple problems into one large problem
- b) Breaking down a complex problem into smaller, manageable parts
- c) Writing code without planning

- d) Designing the user interface

Answer: b

Analysis:

- **Option a)** is incorrect because decomposition is about breaking down problems, not combining them.
- **Option b)** is correct because decomposition helps break down complex problems into smaller, manageable parts.
- **Option c)** is incorrect because decomposition involves planning.
- **Option d)** is incorrect because decomposition is not about UI design.

3. Given the following code snippet:

```
public bool IsEven(int number)
{
    return number % 2 == 0;
}
```

How would you apply abstraction to improve this function?

- a) Combine it with another function to check if a number is odd.
- b) Simplify it by removing the modulus operation.
- c) Create a general-purpose function that checks any number property.
- d) Leave it as it is, since it's already optimal.

Answer: c

Analysis:

- **Option a)** does not apply abstraction.
- **Option b)** is incorrect because removing the modulus operation would not check if the number is even.
- **Option c)** is correct because creating a general-purpose function that checks any number property applies abstraction.
- **Option d)** is incorrect because abstraction can further improve the function.

4. Given the task to find the maximum value in an array, what is the first step according to PADM?

- a) Start coding immediately to find the maximum value.
- b) Analyze the problem to determine the requirements and constraints for finding the maximum value.
- c) Test different arrays to ensure the function works.

- d) Deploy the solution to production.

Answer: b

Analysis:

- **Option a)** is incorrect because starting to code immediately skips important steps.
- **Option b)** is correct because analyzing the problem is the first step according to PADM.
- **Option c)** is incorrect because testing is done after analysis and implementation.
- **Option d)** is incorrect because deployment is the final step.

5. Given the following code snippet:

```
public int FindMax(int[] numbers)
{
    int max = numbers[0];
    for (int i = 1; i < numbers.Length; i++)
    {
        if (numbers[i] > max)
        {
            max = numbers[i];
        }
    }
    return max;
}
```

How would you improve this code snippet to enhance performance and readability?

- a) Add more variables to track the minimum value as well.
- b) Document the method, then review and optimize the loop conditions.
- c) Rewrite the function to use recursion.
- d) Remove the loop and use a built-in method.

Answer: b

Analysis:

- **Option a)** is incorrect because tracking the minimum value is not relevant to finding the maximum value.
- **Option b)** is correct because documenting and optimizing improves readability and performance.
- **Option c)** is incorrect because recursion is not necessary for this task and may reduce performance.
- **Option d)** is incorrect because using a built-in method without understanding it might not always be optimal or applicable.

Collaborative Coding

1. You are starting a new project and want to initialize a Git repository. Which sequence of commands correctly sets up the repository and stages your initial files for commit?

- a)

```
git clone [url]
git add .
git commit -m "Initial commit"
```

- b)

```
git init
git add .
git commit -m "Initial commit"
```

- c)

```
git status
git add .
git commit -m "Initial commit"
```

- d)

```
git init
git commit -m "Initial commit"
git add .
```

Answer: b

Analysis:

- **Option a)** is incorrect because `git clone` is used to copy an existing repository, not to initialize a new one.
- **Option b)** is correct because `git init` initializes a new repository, `git add .` stages all files, and `git commit -m "Initial commit"` commits the changes.
- **Option c)** is incorrect because `git status` only shows the status of the repository and does not initialize or stage files.
- **Option d)** is incorrect because you need to add files before committing.

2. Given the following scenario, which command stages all your modified files for commit?

~~~
```bash
# List of changes
modified:   Contact.cs
~~~

~~~
modified:   Program.cs
```
~~~

- a) `git status`
- b) `git add .`
- c) `git commit -m "Add changes"`
- d) `git init`

Answer: b

Analysis:

- **Option a)** is incorrect because `git status` shows the current status.
- **Option b)** is correct because `git add .` stages all changes.
- **Option c)** is incorrect because `git commit -m "Add changes"` commits staged changes.
- **Option d)** is incorrect because `git init` initializes a new repository.

3. After staging your changes, which command do you use to commit them to your local repository?

- a) `git add .`
- b) `git status`
- c) `git commit -m "Initial commit"`
- d) `git push -m "Initial commit"`

Answer: c

Analysis:

- **Option a)** is incorrect because `git add .` stages changes.
- **Option b)** is incorrect because `git status` shows the current status.
- **Option c)** is correct because `git commit -m "Initial commit"` commits staged changes.
- **Option d)** is incorrect because `git push -m "Initial commit"` is not valid; `git push` is used to push changes to a remote repository without a message.

4. You have modified two files: `Contact.cs` and `Program.cs`. Which sequence of Git commands would you use to stage these files and commit them with the message "Update contact and program logic"?

- a)

```
git add .
git commit -m "Update contact and program logic"
```

- b)

```
git commit -m "Update contact and program logic"
git add .
```

- c)

```
git add Contact.cs Program.cs
git commit -m "Update contact and program logic"
```

- d)

```
git status
git commit -m "Update contact and program logic"
```

Answer: c

Analysis:

- **Option a)** is incorrect because although it stages all changes and commits, it doesn't explicitly show the files being staged.
- **Option b)** is incorrect because you need to add changes before committing.
- **Option c)** is correct because it explicitly stages the modified files and then commits them.
- **Option d)** is incorrect because `git status` only shows the status.

5. Your team has just completed a feature in a feature branch and wants to merge it into the main branch. Before merging, you need to ensure your feature branch is up to date with the main branch. Which sequence of commands accomplishes this?

- a)

```
git checkout main
git merge feature-branch
git push origin main
```

- b)

```
git checkout feature-branch
git pull origin main
git checkout main
git merge feature-branch
```

- c)

```
git pull origin feature-branch
git checkout main
git merge feature-branch
```

- d)

```
git fetch feature-branch
git checkout main
git merge feature-branch
```

Answer: b

Analysis:

- **Option a)** is incorrect because it merges the feature branch into main without ensuring the feature branch is up to date with main.
- **Option b)** is correct because it updates the feature branch with changes from main before merging into main.
- **Option c)** is incorrect because it pulls changes from the feature branch into the current branch.
- **Option d)** is incorrect because `git fetch` does not update the feature branch with changes from main.

Unit Testing

1. Which attribute in xUnit is used to define a test method?

- a) [Test]
- b) [Fact]
- c) [InlineData]
- d) [SetUp]

Answer: b

Analysis:

- **Option a)** `[Test]` is used in NUnit, not xUnit.
- **Option b)** `[Fact]` is the correct attribute used in xUnit to define a test method.
- **Option c)** `[InlineData]` is used with `[Theory]` in xUnit to provide data for parameterized tests, not to define a test method directly.
- **Option d)** `[SetUp]` is used in NUnit, not xUnit.

2. What is the purpose of the Arrange-Act-Assert pattern in unit testing?

- a) To ensure tests are isolated from each other.

- b) To structure tests clearly, separating setup, execution, and verification phases.
- c) To minimize the time needed to write tests.
- d) To ensure tests cover all possible code paths.

Answer: b

Analysis:

- **Option a)** describes test isolation but not the purpose of Arrange-Act-Assert.
- **Option b)** correctly describes the purpose of the pattern, which is to structure tests clearly.
- **Option c)** is not the main purpose of the pattern.
- **Option d)** focuses on coverage, not on the structure of the tests.

3. Which part of a unit test best demonstrates the "Act" step in the Arrange/Act/Assert pattern?

- a) Executing the method under test
- b) Verifying the outcome of the method execution
- c) Initializing the objects required for the test
- d) Examining the behavior of dependencies

Answer: a

Analysis:

- **Option a)** correctly describes the Act step.
- **Option b)** describes the Assert step.
- **Option c)** describes the Arrange step.
- **Option d)** does not specifically describe the Act step.

4. Which xUnit feature would you use to validate the behavior of the following method across a range of inputs?

```csharp
public bool IsPrime(int number)
{
    if (number <= 1) return false;
    for (int i = 2; i < number; i++)
        if (number % i == 0) return false;
    return true;
}
```

- a) [Fact] with multiple methods
- b) [Theory] with [InlineData]
- c) [Fact] with a loop inside the test method
- d) [Fact] with [Theory]

Answer: b

Analysis:

- **Option a)** would require multiple `[Fact]` methods for different inputs.
- **Option b)** correctly uses `[Theory]` with `[InlineData]` to provide multiple inputs for a single test method.
- **Option c)** is incorrect because looping inside the test method is not ideal.
- **Option d)** is incorrect because `[Fact]` does not work with `[Theory]`.

5. Consider the following code snippet:

```
public int Add(int a, int b)
{
    return a + b;
}
```

Which of the following is a correct and ideal xUnit test for the `Add` method following the Arrange-Act-Assert pattern?

- a) Arrange: var a = 1; var b = 2; Act: var result = a + b; Assert: Assert.Equal(3, result);
- b) Arrange: var a = 2; var b = 1; Act: var result = Add(a, b); Assert: Assert.Equal(3, a + b);
- c) Arrange: var a = 2; var b = 2; Act: var result = Add(a, b); Assert: Assert.Equal(4, result);
- d) Arrange: var a = 3; var b = 0; Act: var result = Add(3, 0); Assert: Assert.Equal(3, result);

Answer: c

Analysis:

- **Option a)** is incorrect because it does not call the `Add` method.
- **Option b)** is incorrect because the assertion checks `a + b` instead of the result.
- **Option c)** is correct because it follows the Arrange-Act-Assert pattern correctly.
- **Option d)** is less ideal because it uses magic values directly in the Act step.

6. Which of the following correctly describes the difference between a mock and a stub?

- a) A mock is a controllable replacement for an existing dependency, while a stub decides whether a test has passed or failed.
- b) A mock decides whether a test has passed or failed, while a stub is a controllable replacement for an existing dependency.
- c) A stub is a special type of mock.
- d) There is no difference; the terms are interchangeable.

Answer: b

Analysis:

- **Option a)** incorrectly describes the roles.
- **Option b)** correctly describes the roles: mocks are used to verify interactions, while stubs provide controlled responses.
- **Option c)** is incorrect because a stub is not a special type of mock.
- **Option d)** is incorrect because mocks and stubs have different purposes.

7. Which of the following is an example of a magic string in a unit test?

- a) Assert.Equal(3, result);
- b) var stringCalculator = new StringCalculator();
- c) const string MAXIMUM_RESULT = "1001";
- d) Action actual = () => stringCalculator.Add("1001");

Answer: d

Analysis:

- **Option a)** is a numeric value, not a string.
- **Option b)** is an object instantiation, not a magic string.
- **Option c)** is a constant, not a magic string.
- **Option d)** uses a hard-coded string value in the test, which is an example of a magic string.

8. Which practice should be avoided when writing unit tests?

- a) Using logic in tests
- b) Using the Arrange-Act-Assert pattern
- c) Naming tests with method name, scenario, and expected behavior
- d) Writing minimally passing tests

Answer: a

Analysis:

- **Option a)** is correct because using logic in tests can introduce errors and complexity.
- **Option b)** is incorrect because the Arrange-Act-Assert pattern is a best practice.
- **Option c)** is incorrect because clear naming of tests is a best practice.
- **Option d)** is incorrect because writing minimally passing tests is generally a good practice.

9. Consider the following code:

```
public interface IDateTimeProvider
{
    DayOfWeek DayOfWeek();
}

public class PriceCalculator
{
    public int GetDiscountedPrice(int price, IDateTimeProvider dateTimeProvider)
    {
        if (dateTimeProvider.DayOfWeek() == DayOfWeek.Tuesday)
        {
            return price / 2;
        }
        else
        {
            return price;
        }
    }
}
```

Which of the following tests correctly verifies the `GetDiscountedPrice` method?

- a) Verify that `GetDiscountedPrice` returns full price when not Tuesday by mocking `IDateTimeProvider` to return Monday.
- b) Verify that `GetDiscountedPrice` returns half price when not Tuesday by mocking `IDateTimeProvider` to return Wednesday.
- c) Mock `IDateTimeProvider` to return Monday and verify `GetDiscountedPrice` returns half price.
- d) None of the above

Answer: a

Analysis:

- **Option a)** is correct because it verifies the method's behavior when it is not Tuesday.
- **Option b)** is incorrect because it misrepresents the condition; it should verify full price when not Tuesday.
- **Option c)** is incorrect because it incorrectly expects half price on Monday.
- **Option d)** is incorrect because option a) is correct.

Graphical Desktop Development

1. In the MVVM pattern, what is the primary role of the ViewModel?

- a) To handle all user interactions and UI updates.
- b) Presentation logic and state management.
- c) To directly manipulate the Model and View.
- d) To define the layout and appearance of the UI.

Answer: b

Analysis:

- **Option a)** is incorrect because handling all user interactions and UI updates is more the responsibility of the View.
- **Option b)** is correct because the ViewModel is responsible for presentation logic and state management.
- **Option c)** is incorrect because the ViewModel should not directly manipulate the View.
- **Option d)** is incorrect because defining the layout and appearance of the UI is the responsibility of the View.

2. What is the main purpose of the Model in the MVVM pattern?

- a) To define the structure and layout of the UI.
- b) To handle user inputs and interactions.
- c) To manage data and business logic.
- d) To notify the View of property changes.

Answer: c

Analysis:

- **Option a)** is incorrect because the Model does not define the UI structure and layout.
- **Option b)** is incorrect because handling user inputs and interactions is the role of the View.
- **Option c)** is correct because the Model manages data and business logic.
- **Option d)** is incorrect because it is not the primary purpose of the Model.

3. In the context of MVVM and Avalonia UI, which class is specifically designed for collections and provides automatic notifications to the UI when the underlying data changes?

- a) `BindingMode`
- b) `ObservableCollection`
- c) `ReactiveObject`
- d) `INotifyCollectionChanged`

Answer: b

Analysis:

- **Option a)** is incorrect because `BindingMode` specifies the direction of data binding.
- **Option b)** is correct because `ObservableCollection` is designed for collections that automatically notify the UI of data changes.
- **Option c)** is incorrect because `ReactiveObject` is used for property change notifications, not collections.
- **Option d)** is incorrect because `INotifyCollectionChanged` is an interface, not a class.

4. Which of the following best describes the Observer pattern?

- a) A pattern where objects are created from a template.
- b) A pattern where an object delegates its responsibilities to another object.
- c) A pattern where an object notifies its dependents of state changes.
- d) A pattern where two objects communicate through a shared interface.

Answer: c

Analysis:

- **Option a)** describes the Prototype pattern.
- **Option b)** describes the Delegation pattern.
- **Option c)** is correct because the Observer pattern involves an object notifying its dependents (observers) of state changes.
- **Option d)** is incorrect because the communication is not necessarily through a shared interface.

5. In a common implementation of the Observer pattern, what does the subject typically maintain?

- a) A queue of state change events.
- b) A list of its observers.
- c) A history of all past state changes.
- d) A reference to the most recently updated observer.

Answer: b

Analysis:

- **Option a)** is incorrect because the subject maintains a list of observers, not a queue of events.
- **Option b)** is correct because the subject maintains a list of its observers to notify them of state changes.
- **Option c)** is incorrect because the subject does not typically maintain a history of state changes.
- **Option d)** is incorrect because the subject does not maintain a reference to the most recently updated observer.

6. How does the `ObservableCollection` class support the Observer pattern in Avalonia UI?

- a) It triggers manual updates to the UI when the collection changes.
- b) It automatically updates the UI when the collection changes.
- c) It prevents changes to the collection.
- d) It stores data without notifying the View.

Answer: b

Analysis:

- **Option a)** is incorrect because `ObservableCollection` does not require manual updates.
- **Option b)** is correct because `ObservableCollection` automatically updates the UI when the collection changes.
- **Option c)** is incorrect because `ObservableCollection` allows changes to the collection.
- **Option d)** is incorrect because `ObservableCollection` notifies the View of changes.

7. What is the main advantage of using the MVVM pattern in Avalonia UI development?

- a) Increased coupling of UI and logic
- b) Enhanced data binding performance
- c) Separation of concerns
- d) Reduced XAML complexity

Answer: c

Analysis:

- **Option a)** is incorrect because MVVM aims to reduce coupling.
- **Option b)** is a benefit but not the main advantage.
- **Option c)** is correct because the main advantage of MVVM is the separation of concerns.
- **Option d)** is incorrect because MVVM does not necessarily reduce XAML complexity.

8. In Avalonia UI, what role does `INotifyPropertyChanged` play in the implementation of the Observer pattern?

- a) It handles direct manipulation of UI elements.
- b) It allows objects to notify subscribers about changes to properties.
- c) It manages application state and lifecycle.
- d) It provides static data storage.

Answer: b

Analysis:

- **Option a)** is incorrect because `INotifyPropertyChanged` does not directly manipulate UI elements.
- **Option b)** is correct because `INotifyPropertyChanged` allows objects to notify subscribers about property changes.
- **Option c)** is incorrect because `INotifyPropertyChanged` does not manage application state and lifecycle.
- **Option d)** is incorrect because `INotifyPropertyChanged` does not provide static data storage.

9. In ReactiveUI, what is the primary purpose of the `RaiseAndSetIfChanged` method within a ViewModel?

- a) To automatically bind a property to a UI element.
- b) To trigger an animation when a property value changes.
- c) To update a property's value and notify the UI of the change if the value has actually changed.
- d) To validate user input and display error messages.

Answer: c

Analysis:

- **Option a)** is incorrect because `RaiseAndSetIfChanged` does not handle binding directly.
- **Option b)** is incorrect because `RaiseAndSetIfChanged` does not trigger animations.
- **Option c)** is correct because `RaiseAndSetIfChanged` updates the property's value and notifies the UI if the value changes.
- **Option d)** is incorrect because `RaiseAndSetIfChanged` does not handle validation and error messages.

10. In the context of Avalonia UI and MVVM, what does the Model primarily represent?

- a) The application's business logic and data.
- b) The structure and visual layout of the user interface.
- c) The presentation logic and state of the UI.
- d) The interaction logic between the View and the ViewModel.

Answer: a

Analysis:

- **Option a)** is correct because the Model represents the application's business logic and data.
- **Option b)** is incorrect because the structure and layout of the UI are handled by the View.
- **Option c)** is incorrect because the presentation logic and state of the UI are handled by the ViewModel.
- **Option d)** is incorrect because the interaction logic between the View and the ViewModel is part of the ViewModel's role.

11. In the MVVM pattern, what is the primary responsibility of the ViewModel?

- a) Direct handling of user interface events.
- b) Defining the visual appearance of the application.
- c) Handling presentation logic and state management.
- d) Data persistence management.

Answer: c

Analysis:

- **Option a)** is incorrect because direct handling of UI events is more the role of the View.
- **Option b)** is incorrect because defining the visual appearance is the role of the View.
- **Option c)** is correct because the ViewModel handles presentation logic and state management.
- **Option d)** is incorrect because data persistence management is usually part of the Model's responsibilities.

12. In the MVVM pattern, what is the primary responsibility of the View?

- a) To handle business logic and data manipulation.
- b) To define the structure, layout, and appearance of the UI.
- c) To act as an intermediary between the Model and the ViewModel.
- d) To manage application state and lifecycle.

Answer: b

Analysis:

- **Option a)** is incorrect because business logic and data manipulation are handled by the Model.
- **Option b)** is correct because the View defines the structure, layout, and appearance of the UI.
- **Option c)** is incorrect because the ViewModel acts as the intermediary.
- **Option d)** is incorrect because managing application state and lifecycle is a broader concern.

13. How does the `ReactiveObject` class support the MVVM pattern in Avalonia UI?

- a) It directly handles user inputs.
- b) It manages the application's lifecycle.
- c) It provides a static method to update the UI.
- d) It notifies the View of property changes.

Answer: d

Analysis:

- **Option a)** is incorrect because `ReactiveObject` does not directly handle user inputs.
- **Option b)** is incorrect because `ReactiveObject` does not manage the application's lifecycle.
- **Option c)** is incorrect because `ReactiveObject` does not provide a static method to update the UI.
- **Option d)** is correct because `ReactiveObject` notifies the View of property changes, supporting data binding in the MVVM pattern.

14. How does the Observer pattern promote loose coupling between objects?

- a) The subject and observers have direct references to each other.
- b) The subject and observers communicate through a shared global variable.

- c) The subject and observers interact through a well-defined interface.
- d) The subject and observers are tightly integrated within a single class.

Answer: c

Analysis:

- **Option a)** is incorrect because direct references would create tight coupling.
- **Option b)** is incorrect because using a shared global variable is not a good practice for promoting loose coupling.
- **Option c)** is correct because the Observer pattern promotes loose coupling by having the subject and observers interact through a well-defined interface.
- **Option d)** is incorrect because tight integration within a single class does not promote loose coupling.

15. What is the main advantage of using the Observer pattern?

- a) It improves the performance of the application.
- b) It reduces the memory usage of the application.
- c) It makes the code more complex and harder to understand.
- d) It enables objects to communicate with each other without tight coupling.

Answer: d

Analysis:

- **Option a)** is incorrect because the primary advantage is not related to performance.
- **Option b)** is incorrect because the primary advantage is not related to memory usage.
- **Option c)** is incorrect because the pattern aims to make communication easier, not more complex.
- **Option d)** is correct because the main advantage of the Observer pattern is to enable communication between objects without tight coupling.