Question 1

How do you enable your project to use navigation components?

* Make sure every Activity class extends the class NavigationActivity.
* Use the NavigationController class as the launch Activity.
* Add <uses-navigation> to the Android manifest file.
* Add dependencies for navigation-fragment-ktx and navigation-ui-ktx in the build.gradle (module) file.

Question 2

Where are the possible routes through your app defined?

* In a file (often called navigation.xml) in the **res > layout** folder.
* In a file (often called navigation.xml) in the **app > navigation** folder.
* In a file (often called navigation.xml) in the **res > navigation** folder.
* In the android-manifest.xml file in the <navigation> element.

Question 3

Which of the following statements about the NavHostFragment are true? Select all that apply.

* As the user moves between destinations defined in the navigation graph, the NavHostFragment swaps the fragments in and out as necessary.
* You can click the NavHostFragment in the Project view to open the navigation graph.
* You add the NavHostFragment to the main layout by adding a <fragment> whose name is androidx.navigation.fragment.NavHostFragment.
* You must create a single NavHostFragment subclass and implement the onNavigate() method to handle different kinds of navigation (such as button clicks).

Question 4

Which of the following statements about the navigation graph are true? Select all that apply.

* You create a potential path through the app from one Fragment to another by connecting the fragments in the navigation graph.
* The type of a connection between fragments is Action.
* You must add the type="navigation" attribute to every <fragment> that is included in the navigation graph.
* To open the navigation graph, you double-click the navigation file (navigation.xml) in the Android Studio Project pane, then click the **Design** tab.

Question 5

Where do you set the ID of a Fragment to be used in navigation?

* In the Fragment's layout file, either by setting the ID attribute in the design editor or in the layout XML file in the **res > layout** folder.
* In the project's navigation file, either by setting the ID attribute in the navigation graph or in the navigation XML file in the **res > navigation** folder.
* You need to set the ID in both the navigation file for the app and the layout file for the Fragment.
* Set the ID variable in the relevant Fragment class.

Question 6

The News app has a NewsFragment that displays a **Show headlines** button. The goal is that when the user clicks this button, the app navigates to the HeadlinesFragment.

Assume that you've added a connection from the NewsFragment to the HeadlinesFragment in the navigation graph, as shown here:

Diagram

Description automatically generated with medium confidence

What else do you need to do so that when the user taps the **Show headlines** button, the app navigates to the HeadlinesFragment?

* Actually, you don't need to do anything else. It's enough to set the navigation paths in the navigation graph.
* In the onclickListener for the **Show headlines** button, call navigate() on the navigation controller, passing the class name of the destination Fragment (in this case HeadlinesFragment) .
* In the onclickListener for the **Show headlines** button, call navigate() on the navigation controller, passing the action that connects the NewsFragment to the HeadlinesFragment.
* In the onclickListener for the **Show headlines** button, call navigateTo() on the container Fragment, passing the class name of the destination Fragment (in this case HeadlinesFragment) .

Question 7

When users navigate through an app, sometimes they want to retrace their steps back through the screens they have already visited.

Assume the following:

* fragmentA is connected to fragmentB by action\_FragmentA\_to\_FragmentB.
* fragmentB is connected to fragmentC by action\_FragmentB\_to\_FragmentC

A picture containing box and whisker chart

Description automatically generated

Which of the following statements are true regarding navigating forward and backward through the app? (Choose all that apply.)

* The destination of the action\_FragmentA\_to\_FragmentB action specifies that when the user is at FragmentA, the next destination in the app is FragmentB.
* The destination of the action\_FragmentA\_to\_FragmentB action sets the next destination that the user goes to, whether the user taps a button in the app or taps the Back button at the bottom of the screen.
* The popUpTo attribute of the action can modify where the app navigates to if the user taps the system Back button.
* The popUpTo attribute of the action can modify where the user goes next as they navigate forward through the app.

Question 8

When users navigate through an app, sometimes they want to retrace their steps back through the screens they have already visited. However, you can use the popUpTo and popUpToInclusive attributes of an action to modify the path backward through the app.

Assume the following:

* fragmentA is connected to fragmentB by action\_FragmentA\_to\_FragmentB.
* fragmentB is connected to fragmentC by action\_FragmentB\_to\_FragmentC.

A picture containing box and whisker chart

Description automatically generated

The user navigates from fragmentA to fragmentB to fragmentC, then taps the system Back button. In this situation, let's say you want the app to navigate back to fragmentA (instead of fragmentB). What's the correct way to set the popUpTo and popUpToInclusive attributes?

* For action\_FragmentA\_to\_FragmentB, set popUpTo to fragmentB and popUpToInclusive to no value. For action\_FragmentB\_to\_FragmentC, set popUpTo to fragmentA and popUpToInclusive to true.
* For action\_FragmentA\_to\_FragmentB, set popUpTo to fragmentA and popUpToInclusive to true. For action\_FragmentB\_to\_FragmentC, set popUpTo to fragmentA and popUpToInclusive to true.
* For action\_FragmentA\_to\_FragmentB, set popUpTo to none and popUpToInclusive to no value. (You can omit both attributes.) For action\_FragmentB\_to\_FragmentC, set popUpTo to fragmentA and popUpToInclusive to true.
* For action\_FragmentA\_to\_FragmentB, set popUpTo to none and popUpToInclusive to no value. (You can omit both attributes.) For action\_FragmentB\_to\_FragmentC, set popUpTo to fragmentA and popUpToInclusive to false.

Question 9

Assume that the action action\_headlinesFragment\_to\_newsArticle in the destination graph has a popUpTo value of newsFragment:

Diagram

Description automatically generated

Assume that the user opens the app and navigates through the screens in the following sequence (without using the Back button):

Open app into News home > Headlines > News details

When the user is viewing the **News detail** screen, what happens If they tap the system Back button at the bottom of the screen?

Select all that apply (remembering that popUpTo is newsFragment).

* If popUpToInclusive is true: Android navigates out of the app because there is nothing left in the back stack for this app.
* If popUpToInclusive is false: The app goes back to the news home screen.
* If popUpToInclusive is true: The app goes back to the news home screen.
* If popUpToInclusive is false: The app goes back to the headlines screen.

Question 10

Where do you define the items for a menu?

* It depends on where the menu will be shown. For a navigation drawer menu, add an <item> tag for each menu item in the *menu*.xml file in **res > drawer** folder. For the options menu, add an <item> tag for each menu item in the *menu*.xml file in **res > options** folder.
* In the layout file for the Fragment or Activity that displays the menu, add a <menu> tag that contains <item> tags for each item.
* In a *menu\_name*.xml file in the **res > menu** folder, add an <item> tag for each menu item. Create separate XML files for each separate menu.
* In the android\_manifest.xml file, add a <menus> tag that contains a <menu> tag for each menu. For each <menu> tag, add an <item> tag for each menu item.

Question 11

To enable the options menu in your app, you need to define the menu items. Then what do you need to do in the Activity or Fragment where the options menu is to appear?

Select all that apply:

* Call setHasOptionsMenu(true) in onCreate() for an Activity, or in onCreateView() for a Fragment.
* Implement onCreateOptionsMenu() in the Activity or Fragment to create the menu.
* Set the onClick attribute in the menu's XML file to onShowOptionsMenu, unless you are implementing a custom onClick listener for the options menu, in which case specify the name of the custom onClick listener instead.
* Implement onOptionsItemSelected() in the Activity or Fragment to determine what happens when a user selects a menu item in the options menu.

Question 12

What do you need to do to enable a navigation drawer in your app? You can assume that your project is using the navigation graph and that you've already defined the menu items.

Select all that apply:

* Use <DrawerLayout> as the root view in the relevant layout file, and add a <NavigationView> tag to that layout.
* Use <Navigation> as the root view in the relevant layout file, and add a <NavigationView> tag to that layout.
* In the <NavigationView> in the layout, set the android:menu attribute to the navigation drawer menu.
* In the navigation XML file, make sure that the navigation menu has an ID.

**Question 13**

Following on from the previous question, you need to write some code to enable the navigation drawer to be displayed when the user swipes in from the left side of the screen?

In onCreate() within the Activity that creates the navigation controller, what is the right code to add?

* A:

NavigationUI.setupWithNavController(  
    navigationLayoutID, navigationMenuID)

* B:

NavigationUI.setupWithNavController(  
    navigationView, navigationController)

* C:

NavigationDrawer.setupWithNavInterface(  
    navigationView, navigationController)

* D:

NavigationDrawer.setupWithNavController(  
    navigationView, navigationMenuID)

Question 14

How do you add support for the Up button at the top of the screen to enable users to get back to the app's home screen from anywhere in the app? What do you need to do in the relevant Activity?

Select all that apply:

* In the res > menu folder, create the up\_menu.xml file.
* Link the navigation controller to the app bar using NavigationUI.setupActionBarWithNavController(context,navigationController)
* Override onSupportNavigateUp() method to call navigateUp() on the navigation controller.
* In the navigation graph, make sure the starting Fragment has an ID of HomeFragment.