1) ATM testing scenario

1. Cash Dispensing: Check if the ATM is dispensing the correct amount of cash - as much as user requested. Verify that the ATM is dispensing the appropriate denominations of currency and that the banknotes are not damaged or torn.
2. Card Reader: Ensure that the ATM's card reader is functioning correctly by testing different types of cards: debit cards, credit cards, cards from different banks. Make sure that the card is accepted and that the PIN is validated.
3. Deposits: Test the ATM's deposit function by accepting cash and ensuring that the ATM is accurately counting the amount and adding it to the user's account.
4. Transaction History: Verify that the ATM is recording transaction history accurately, including dispensing cash, deposits, and transfers.
5. Receipts: Check that the ATM is printing receipts accurately and that the receipts contain all necessary information, including transaction details, balances, and other relevant information.
6. Network Connection: Verify that the ATM is properly connected to the network and that it can communicate with the bank's servers.
7. Security: Test the ATM's security features by attempting to get cash without a valid card or with incorrect PIN, check if it blocks the card if PIN is entered incorrectly for several times. When depositing cash beware for counterfeit or damaged money. Also, verify that the ATM is equipped with the necessary security measures, including a camera, a card skimming detector, and other features to prevent crime actions.

Testing should be performed from the user's perspective, ensuring that the ATM is functioning as expected and providing a good user experience.

2) Test scenario for application performing the four basic functions: addition, subtraction, multiplication, division:

1. Test for accuracy: test by entering different sets of numbers and ensuring that the calculated results are correct by comparing the calculated results with the expected results.
2. Test for input validation: test by entering invalid inputs, such as letters, symbols, and characters that are not numbers, to see how the application handles them. The application should reject these inputs or return an error message.
3. Test for edge cases: test edge cases, such as dividing by zero or using very large or small numbers, to ensure that the application can handle these cases appropriately.
4. Test for order of operations: Test the application for order of operations by entering expressions that involve multiple operations, such as 5 + 2 * 3, to verify that the application is performing the operations in the correct order.
5. Test for rounding: test for rounding by entering numbers with decimal places and verifying that the application is rounding the results correctly.
6. Test for negative numbers: test by entering expressions that involve negative numbers, such as -5 + 3.
7. Test for performance: test by entering a large number of expressions and verifying that the application is responding quickly and not crashing.