```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract CombinedFunctions
{
   uint public storedValue;

   // Constructor to initialize storedValue
   constructor()
   {
      storedValue = 10;  // Initialize storedValue with 10
   }

   // View function to check the stored value
   function getStoredValue() public view returns (uint)
   {
      return storedValue;
   }

   // Pure function to add two numbers
   function addNumbers(uint a, uint b) public pure returns (uint)
   {
      return a + b;
   }

   // Payable function to receive Ether and update storedValue
   function receiveEtherAndUpdateValue(uint newValue) public payable
   {
      storedValue = storedValue+newValue;
   }
}
```

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract VisibilityExample
{
   uint public publicValue = 10;
   uint internal internalValue = 20;
   uint private privateValue = 30;

   // Public function: Can be called externally and internally.
   function setPublicValue(uint newValue) public
   {
      publicValue = newValue;
   }
   // Internal function: Can only be called within the contract or derived
contracts.
   function updateInternalValue(uint newValue) internal
   {
      internalValue = newValue;
   }
   // Private function: Can only be called within this contract.
   function updatePrivateValue(uint newValue) private
   {
      privateValue = newValue;
   }

   // Public function that uses internal and private functions.
   function modifyValues(uint newInternalValue, uint newPrivateValue) public
   {
      updateInternalValue(newInternalValue); // Calls internal function
      updatePrivateValue(newPrivateValue); // Calls private function
   }

   // Public function to read internal and private variables
   function getInternalAndPrivateValues() public view returns (uint, uint)
   {
      return (internalValue, privateValue); // Reading internal and private
variables.
   }
}
```

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract ParentContract
{
   uint public parentValue;
   constructor()
   {
      parentValue = 100;  // Setting a default value in the parent contract
   }

   function setParentValue(uint newValue) public
   {
      parentValue = newValue;
   }

   function getParentValue() public view returns (uint)
   {
      return parentValue;
   }
}

contract ChildContract is ParentContract
{
   function updateParentValue(uint newValue) public
   {
      setParentValue(newValue);  // Calling the parent contract's function
   }
}
```

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract RequireAndRevertExample
{
   uint public balance;
   constructor()
   {
      balance = 100; // Starting balance of 100
   }

   function withdraw(uint amount) public
   {
      require(amount <= balance, "Insufficient balance to withdraw");
      balance -= amount;
   }

   function deposit(uint amount) public
   {
      balance += amount;
   }

   function riskyOperation(uint riskAmount) public
   {
      if (riskAmount > 50)
      {
         revert("Operation too risky. Aborting transaction.");
      }

      balance -= riskAmount;
   }
}
```