```solidity
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v4.8.0) (utils/Address.sol)

pragma solidity ^0.8.1;

/**
 * @dev Collection of functions related to the address type
 */
library Address {
    /**
     * @dev Returns true if `account` is a contract.
     *
     * [IMPORTANT]
     * ====
     * It is unsafe to assume that an address for which this function returns
     * false is an externally-owned account (EOA) and not a contract.
     *
     * Among others, `isContract` will return false for the following
     * types of addresses:
     *
     *  - an externally-owned account
     *  - a contract in construction
     *  - an address where a contract will be created
     *  - an address where a contract lived, but was destroyed
     * ====
     *
     * [IMPORTANT]
     * ====
     * You shouldn't rely on `isContract` to protect against flash loan attacks!
     *
     * Preventing calls from contracts is highly discouraged. It breaks composability, breaks support for smart wallets
     * like Gnosis Safe, and does not provide security since it can be circumvented by calling from a contract
     * constructor.
     * ====
     */
    function isContract(address account) internal view returns (bool) {
        // This method relies on extcodesize/address.code.length, which returns 0
        // for contracts in construction, since the code is only stored at the end
        // of the constructor execution.

        return account.code.length > 0;
    }

    /**
     * @dev Replacement for Solidity's `transfer`: sends `amount` wei to
     * `recipient`, forwarding all available gas and reverting on errors.
     *
     * https://eips.ethereum.org/EIPS/eip-1884[EIP1884] increases the gas cost
     * of certain opcodes, possibly making contracts go over the 2300 gas limit
     * imposed by `transfer`, making them unable to receive funds via
     * `transfer`. {sendValue} removes this limitation.
     *
     * https://consensys.net/diligence/blog/2019/09/stop-using-soliditys-transfer-now/[Learn more].
     *
     * IMPORTANT: because control is transferred to `recipient`, care must be
     * taken to not create reentrancy vulnerabilities. Consider using
     * {ReentrancyGuard} or the
     * https://solidity.readthedocs.io/en/v0.5.11/security-considerations.html#use-the-checks-effects-interactions-pattern[checks-effects-interactions pattern].
     */
    function sendValue(address payable recipient, uint256 amount) internal {
        require(address(this).balance >= amount, "Address: insufficient balance");

        (bool success, ) = recipient.call{value: amount}("");
        require(success, "Address: unable to send value, recipient may have reverted");
    }

```

```
67        /**
68         * @dev Performs a Solidity function call using a low level `call`. A
69         * plain `call` is an unsafe replacement for a function call: use this
70         * function instead.
71         *
72         * If `target` reverts with a revert reason, it is bubbled up by this
73         * function (like regular Solidity function calls).
74         *
75         * Returns the raw returned data. To convert to the expected return value,
76         * use
           https://solidity.readthedocs.io/en/latest/units-and-global-variables.html?highlig
           ht=abi.decode#abi-encoding-and-decoding-functions[`abi.decode`].
77         *
78         * Requirements:
79         *
80         * - `target` must be a contract.
81         * - calling `target` with `data` must not revert.
82         *
83         * _Available since v3.1._
84         */
85        function functionCall(address target, bytes memory data) internal returns (bytes
          memory) {
86            return functionCallWithValue(target, data, 0, "Address: low-level call failed"
              );
87        }
88
89        /**
90         * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`], but
           with
91         * `errorMessage` as a fallback revert reason when `target` reverts.
92         *
93         * _Available since v3.1._
94         */
95        function functionCall(
96            address target,
97            bytes memory data,
98            string memory errorMessage
99        ) internal returns (bytes memory) {
100           return functionCallWithValue(target, data, 0, errorMessage);
101       }
102
103       /**
104        * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],
105        * but also transferring `value` wei to `target`.
106        *
107        * Requirements:
108        *
109        * - the calling contract must have an ETH balance of at least `value`.
110        * - the called Solidity function must be `payable`.
111        *
112        * _Available since v3.1._
113        */
114       function functionCallWithValue(
115           address target,
116           bytes memory data,
117           uint256 value
118       ) internal returns (bytes memory) {
119           return functionCallWithValue(target, data, value, "Address: low-level call
              with value failed");
120       }
121
122       /**
123        * @dev Same as
           {xref-Address-functionCallWithValue-address-bytes-uint256-}[`functionCallWithValu
           e`], but
124        * with `errorMessage` as a fallback revert reason when `target` reverts.
125        *
126        * _Available since v3.1._
127        */
128       function functionCallWithValue(
129           address target,
130           bytes memory data,
131           uint256 value,
```

```
132             string memory errorMessage
133         ) internal returns (bytes memory) {
134             require(address(this).balance >= value, "Address: insufficient balance for
                call");
135             (bool success, bytes memory returndata) = target.call{value: value}(data);
136             return verifyCallResultFromTarget(target, success, returndata, errorMessage);
137         }
138 
139         /**
140          * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],
141          * but performing a static call.
142          *
143          * _Available since v3.3._
144          */
145         function functionStaticCall(address target, bytes memory data) internal view
        returns (bytes memory) {
146             return functionStaticCall(target, data, "Address: low-level static call
                failed");
147         }
148 
149         /**
150          * @dev Same as {xref-Address-functionCall-address-bytes-string-}[`functionCall`],
151          * but performing a static call.
152          *
153          * _Available since v3.3._
154          */
155         function functionStaticCall(
156             address target,
157             bytes memory data,
158             string memory errorMessage
159         ) internal view returns (bytes memory) {
160             (bool success, bytes memory returndata) = target.staticcall(data);
161             return verifyCallResultFromTarget(target, success, returndata, errorMessage);
162         }
163 
164         /**
165          * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],
166          * but performing a delegate call.
167          *
168          * _Available since v3.4._
169          */
170         function functionDelegateCall(address target, bytes memory data) internal returns
        (bytes memory) {
171             return functionDelegateCall(target, data, "Address: low-level delegate call
                failed");
172         }
173 
174         /**
175          * @dev Same as {xref-Address-functionCall-address-bytes-string-}[`functionCall`],
176          * but performing a delegate call.
177          *
178          * _Available since v3.4._
179          */
180         function functionDelegateCall(
181             address target,
182             bytes memory data,
183             string memory errorMessage
184         ) internal returns (bytes memory) {
185             (bool success, bytes memory returndata) = target.delegatecall(data);
186             return verifyCallResultFromTarget(target, success, returndata, errorMessage);
187         }
188 
189         /**
190          * @dev Tool to verify that a low level call to smart-contract was successful,
             and revert (either by bubbling
191          * the revert reason or using the provided one) in case of unsuccessful call or
             if target was not a contract.
192          *
193          * _Available since v4.8._
194          */
195         function verifyCallResultFromTarget(
196             address target,
197             bool success,
```

```
198             bytes memory returndata,
199             string memory errorMessage
200         ) internal view returns (bytes memory) {
201             if (success) {
202                 if (returndata.length == 0) {
203                     // only check isContract if the call was successful and the return
                        data is empty
204                     // otherwise we already know that it was a contract
205                     require(isContract(target), "Address: call to non-contract");
206                 }
207                 return returndata;
208             } else {
209                 _revert(returndata, errorMessage);
210             }
211         }
212 
213         /**
214          * @dev Tool to verify that a low level call was successful, and revert if it
             wasn't, either by bubbling the
215          * revert reason or using the provided one.
216          *
217          * _Available since v4.3._
218          */
219         function verifyCallResult(
220             bool success,
221             bytes memory returndata,
222             string memory errorMessage
223         ) internal pure returns (bytes memory) {
224             if (success) {
225                 return returndata;
226             } else {
227                 _revert(returndata, errorMessage);
228             }
229         }
230 
231         function _revert(bytes memory returndata, string memory errorMessage) private pure
             {
232             // Look for revert reason and bubble it up if present
233             if (returndata.length > 0) {
234                 // The easiest way to bubble the revert reason is using memory via
                    assembly
235                 /// @solidity memory-safe-assembly
236                 assembly {
237                     let returndata_size := mload(returndata)
238                     revert(add(32, returndata), returndata_size)
239                 }
240             } else {
241                 revert(errorMessage);
242             }
243         }
244     }
245 
```