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

pragma solidity ^0.8.0;

import "./math/Math.sol";

/**
 * @dev String operations.
 */
library Strings {
    bytes16 private constant _SYMBOLS = "0123456789abcdef";
    uint8 private constant _ADDRESS_LENGTH = 20;

    /**
     * @dev Converts a `uint256` to its ASCII `string` decimal representation.
     */
    function toString(uint256 value) internal pure returns (string memory) {
        unchecked {
            uint256 length = Math.log10(value) + 1;
            string memory buffer = new string(length);
            uint256 ptr;
            /// @solidity memory-safe-assembly
            assembly {
                ptr := add(buffer, add(32, length))
            }
            while (true) {
                ptr--;
                /// @solidity memory-safe-assembly
                assembly {
                    mstore8(ptr, byte(mod(value, 10), _SYMBOLS))
                }
                value /= 10;
                if (value == 0) break;
            }
            return buffer;
        }
    }

    /**
     * @dev Converts a `uint256` to its ASCII `string` hexadecimal representation.
     */
    function toHexString(uint256 value) internal pure returns (string memory) {
        unchecked {
            return toHexString(value, Math.log256(value) + 1);
        }
    }

    /**
     * @dev Converts a `uint256` to its ASCII `string` hexadecimal representation
     with fixed length.
     */
    function toHexString(uint256 value, uint256 length) internal pure returns (string
    memory) {
        bytes memory buffer = new bytes(2 * length + 2);
        buffer[0] = "0";
        buffer[1] = "x";
        for (uint256 i = 2 * length + 1; i > 1; --i) {
            buffer[i] = _SYMBOLS[value & 0xf];
            value >>= 4;
        }
        require(value == 0, "Strings: hex length insufficient");
        return string(buffer);
    }

    /**
     * @dev Converts an `address` with fixed length of 20 bytes to its not
     checksummed ASCII `string` hexadecimal representation.
     */
    function toHexString(address addr) internal pure returns (string memory) {
        return toHexString(uint256(uint160(addr)), _ADDRESS_LENGTH);
    }
}
```