```solidity
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts v4.4.1 (access/IAccessControl.sol)

pragma solidity ^0.8.0;

/**
 * @dev External interface of AccessControl declared to support ERC165 detection.
 */
interface IAccessControl {
    /**
     * @dev Emitted when `newAdminRole` is set as ``role``'s admin role, replacing `previousAdminRole`
     *
     * `DEFAULT_ADMIN_ROLE` is the starting admin for all roles, despite
     * {RoleAdminChanged} not being emitted signaling this.
     *
     * _Available since v3.1._
     */
    event RoleAdminChanged(bytes32 indexed role, bytes32 indexed previousAdminRole, bytes32 indexed newAdminRole);

    /**
     * @dev Emitted when `account` is granted `role`.
     *
     * `sender` is the account that originated the contract call, an admin role
     * bearer except when using {AccessControl-_setupRole}.
     */
    event RoleGranted(bytes32 indexed role, address indexed account, address indexed sender);

    /**
     * @dev Emitted when `account` is revoked `role`.
     *
     * `sender` is the account that originated the contract call:
     *   - if using `revokeRole`, it is the admin role bearer
     *   - if using `renounceRole`, it is the role bearer (i.e. `account`)
     */
    event RoleRevoked(bytes32 indexed role, address indexed account, address indexed sender);

    /**
     * @dev Returns `true` if `account` has been granted `role`.
     */
    function hasRole(bytes32 role, address account) external view returns (bool);

    /**
     * @dev Returns the admin role that controls `role`. See {grantRole} and
     * {revokeRole}.
     *
     * To change a role's admin, use {AccessControl-_setRoleAdmin}.
     */
    function getRoleAdmin(bytes32 role) external view returns (bytes32);

    /**
     * @dev Grants `role` to `account`.
     *
     * If `account` had not been already granted `role`, emits a {RoleGranted}
     * event.
     *
     * Requirements:
     *
     * - the caller must have ``role``'s admin role.
     */
    function grantRole(bytes32 role, address account) external;

    /**
     * @dev Revokes `role` from `account`.
     *
     * If `account` had been granted `role`, emits a {RoleRevoked} event.
     *
     * Requirements:
     *
     * - the caller must have ``role``'s admin role.
```

```
70          */
71         function revokeRole(bytes32 role, address account) external;
72
73         /**
74          * @dev Revokes `role` from the calling account.
75          *
76          * Roles are often managed via {grantRole} and {revokeRole}: this function's
77          * purpose is to provide a mechanism for accounts to lose their privileges
78          * if they are compromised (such as when a trusted device is misplaced).
79          *
80          * If the calling account had been granted `role`, emits a {RoleRevoked}
81          * event.
82          *
83          * Requirements:
84          *
85          * - the caller must be `account`.
86          */
87         function renounceRole(bytes32 role, address account) external;
88     }
89
```