DAYSTAR UNIVERSITY

SCHOOL OF SCIENCE, ENGINEERING AND HEALTH

PROJECT TITLE:WORKPLACE CLEARANCE SYSTEM

LAVENDER ILA

WORKPLACE CLEARANCE SYSTEM

INTRODUCTION

This document explains the Workplace Clearance System implemented in C. The system manages staff and their clearance levels, and processes clearance requests. The following sections will break down the code and its functionality.

CONSTANTS

The following constants are defined to set the limits and constraints for the system:

1. MAX_STAFF: The maximum number of staff members that can be managed (150).
2. NAME_LENGTH: The maximum length of a staff member's name (50 characters).
3. CLEARANCE_LEVELS: The range of clearance levels (1 to 15).
4. MAX_REQUESTS: The maximum number of clearance requests that can be handled (50).

STRUCTS

Two structs are defined to represent staff members and clearance requests:

1. Staff: Contains an ID, name, and clearance level.
2. ClearanceRequest: Contains the staff ID, requested clearance level, and request status.

FUNCTIONS

addStaff

This function adds a new staff member to the system. It checks if the maximum staff limit is reached, then prompts the user to enter the staff member's name and clearance level. The new staff member is assigned a unique ID and added to the array of staff members.

removeStaff

This function removes a staff member by their ID. It searches for the staff member in the array, removes them by shifting subsequent elements, and decrements the staff count. If the staff member is not found, it notifies the user.

viewStaffs

This function displays all staff members in the system. If there are no staff members, it notifies the user.

requestClearance

This function allows a staff member to request a clearance level. It checks if the maximum number of requests is reached, then creates a new request with a 'pending' status and adds it to the array of requests.

viewRequests

This function displays all clearance requests. If there are no requests, it notifies the user.

processRequest

This function processes a clearance request by its ID. It updates the status of the request to 'approved' or 'rejected'. If the request ID is not found, it notifies the user.

MENU

The menu function presents a user interface for interacting with the system. It provides options to:

1. Add a staff member
2. Remove a staff member
3. View all staff members
4. Request clearance
5. View all clearance requests
6. Process a clearance request
7. Exit the program

Based on the user's choice, the appropriate function is called.

OBJECTIVE

The main goal of creating a workplace clearance system is because if an individual would like to transfer into another workplace it will save time. There is a need for an online system instead of a paper based way. It promotes transparency, it makes the process efficient because you do it from the comfort of wherever you are, anytime, also help in future evidence reference.

Below are some specific goals:

- Enhances security thus limiting access to sensitive data and workplace areas to approved personnel.
- Reduces unauthorized access thus decreasing the risk of unauthorized access to crucial areas or information by allowing only those with appropriate clearance levels to enter.
- Increases efficiency reducing the time required to grant or deny clearance requests.
- Track and manage staff access levels and clearance requests effectively.
- Generate reports on clearance levels and request statuses for oversight and review.

THE CODE FOR THE WORKPLACE CLEARANCE SYSTEM

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

//Declare named expressions
#define MAX_STAFF 150
#define NAME_LENGTH 50
#define CLEARANCE_LEVELS 15
#define MAX_REQUESTS 50

//Create new name for existing data types
typedef struct {
    int id;
    char name[NAME_LENGTH];
    int clearanceLevel;
} Staff;

//Create new name for existing data types
typedef struct {
    int staffId;
    int clearanceLevel;
    char status[10];
    // "pending", "approved", or "rejected"
} ClearanceRequest;

void addStaff(Staff staff[], int *count) {
    if (*count >= MAX_STAFF) {
        printf("Maximum staff limit reached.\n");
        return;
    }

    Staff newStaff;
    newStaff.id = *count + 1;

    printf("Enter name: ");
    scanf("%s", newStaff.name);

    printf("Enter clearance level (1-%d): ", CLEARANCE_LEVELS);
    scanf("%d", &newStaff.clearanceLevel);

    staff[*count] = newStaff;
    (*count)++;
    printf("Staff added successfully!\n");
}
```

```
void removeStaff(Staff staffs[], int *count, int id) {
    int found = 0;

    for (int i = 0; i < *count; i++) {
        if (staffs[i].id == id) {
            found = 1;
            for (int j = i; j < *count - 1; j++) {
                staffs[j] = staffs[j + 1];
            }
            (*count)--;
            printf("Staff removed successfully!\n");
            break;
        }
    }

    if (!found) {
        printf("Staff not found.\n");
    }
}

void viewStaffs(Staff staff[], int count) {
    if (count == 0) {
        printf("No staffs to display.\n");
        return;
    }

    for (int i = 0; i < count; i++) {
        printf("ID: %d, Name: %s, Clearance Level: %d\n", staff[i].id, staff[i].name, staff[i].clearanceLevel);
    }
}

void requestClearance(ClearanceRequest requests[], int *count, Staff staff[], int staffId, int clearanceLevel) {
    if (*count >= MAX_REQUESTS) {
        printf("Maximum clearance requests reached.\n");
        return;
    }

    ClearanceRequest newRequest;
    newRequest.staffId = staffId;
    newRequest.clearanceLevel = clearanceLevel;
    strcpy(newRequest.status, "pending");

    requests[*count] = newRequest;
    (*count)++;
    printf("Clearance request submitted successfully!\n");
}
```

```c
void viewRequests(ClearanceRequest requests[], int count) {
    if (count == 0) {
        printf("No clearance requests to display.\n");
        return;
    }

    for (int i = 0; i < count; i++) {
        printf("STAFF ID: %d, Clearance Level: %d, Status: %s\n", requests[i].staffId,
requests[i].clearanceLevel, requests[i].status);
    }
}

void processRequest(ClearanceRequest requests[], int *count, int requestId, char *status) {
    if (requestId < 0 || requestId >= *count) {
        printf("Clearance request not found.\n");
        return;
    }

    strcpy(requests[requestId].status, status);
    printf("Clearance request %d updated to %s.\n", requestId, status);
}

void menu() {
    Staff staffs[MAX_STAFF];
    int staffCount = 0;
    ClearanceRequest requests[MAX_REQUESTS];
    int requestCount = 0;
    int choice, id, clearanceLevel, requestId;
    char status[10];

    while (1) {
        printf("\n  *-----------------------------------------------------------------------* \n");
        printf("\n  *--------------------- WELCOME TO MAMBO LEO COMPANY ---------------------* \n");
        printf("\n  *-----------------------------------------------------------------------* \n");
        printf("\n   |********************** Menu: ************************|\n");
        printf("        1. Add Staff\n");
        printf("        2. Remove Staff\n");
        printf("        3. View Staffs\n");
        printf("        4. Request Clearance\n");
        printf("        5. View Clearance Requests\n");
        printf("        6. Process Clearance Request\n");
        printf("        7. Exit\n");
        printf("\n\nEnter your choice: ");
        scanf("%d", &choice);

        //Switch Statement
        switch (choice) {
            case 1:
                addStaff(staffs, &staffCount);
                break;
            case 2:
                printf("Enter staff ID to remove: ");
                scanf("%d", &id);
                removeStaff(staffs, &staffCount, id);
                break;
            case 3:
                viewStaffs(staffs, staffCount);
                break;
            case 4:
                printf("Enter staff ID: ");
                scanf("%d", &id);
                printf("Enter clearance level (1-%d): ", CLEARANCE_LEVELS);
                scanf("%d", &clearanceLevel);
                requestClearance(requests, &requestCount, staffs, id, clearanceLevel);
                break;
            case 5:
                viewRequests(requests, requestCount);
                break;
            case 6:
                printf("Enter request ID to process: ");
                scanf("%d", &requestId);
                printf("Enter new status (approved/rejected): ");
                scanf("%s", status);
                processRequest(requests, &requestCount, requestId, status);
                break;
            case 7:
                exit(0);
            default:
                printf("Please select a valid option and attempt again.\n");
        }
    }
}

int main() {
    menu();
    return 0;
}
```

CONCLUSION

This Workplace Clearance System provides a basic framework for managing staff and their clearance requests. It can be extended and modified to include more features and handle more complex scenarios as needed.