//Credits By: Clint Eroll A. Capondag

#include <iostream>

#include <string>

#include <conio.h>

#include <cstring>

#include <limits>

using namespace std;

const int MAX\_NAMES = 100;

char globalPassword[10] = "admin123";

void passHead(){

    cout << "\n\n\n\n\t\t\t\t====== COMPROG11 PASSWORD SECURITY v5.0 =======\n";

}

void head(){

    cout << "\t\t\t\t\t      University of Cebu\n";

    cout << "\t\t\t\t\t   Sanciangko St., Cebu City\n";

    cout << "\t\t\t\t\t  College of Computer Studies\n";

}

void getPassword(char \*password) {

    char ch;

    int i = 0;

    while (1) {

        ch = \_getch();

        if (ch == 13) {

            password[i] = '\0';

            break;

        } else if (ch == 8) {

            if (i > 0) {

                printf("\b \b");

                i--;

            }

        } else {

            password[i] = ch;

            printf("\*");

            i++;

        }

    }

}

// case 1

void getNameWithId(string names[], int& size) {

    if (size < MAX\_NAMES) {

        int numberOfNames;

        while (true) {

            head();

            cout << "\n\n\n\n\t\t\t\t\t\t  ADD RECORD\n";

            cout << "\n\n\n\tEnter the number of names you want to add: ";

            if (cin >> numberOfNames && numberOfNames > 0) {

                break;

            } else {

                cout << "\tInvalid input. Please enter a valid number.\n";

                cin.clear();

                cin.ignore(numeric\_limits<streamsize>::max(), '\n');

                cout << "\n\n\tPress any key to continue...";

                getch();

                system("cls");

            }

        }

        system("cls");

        for (int i = 0; i < numberOfNames; ++i) {

            head();

            cout << "\n\n\n\n\t\t\t\t\t\t  ADD RECORD\n";

            string name;

            cout << "\n\n\n\tEnter name   : ";

            cin.ignore();

            getline(cin, name);

            int id;

            cout << "\tEnter ID no. : ";

            cin >> id;

            bool isDuplicate = false;

            for (int j = 0; j < size; ++j) {

                int storedId = stoi(names[j].substr(0, names[j].find(" ")));

                string storedName = names[j].substr(names[j].find(" ") + 1);

                if (id == storedId || name == storedName) {

                    isDuplicate = true;

                    cout << "\n\tName or ID already exists. Enter a unique name and ID.\n";

                    cout << "\n\n\tPress any key to continue...";

                    getch();

                    system("cls");

                    --i;

                    break;

                }

            }

            if (isDuplicate) {

                continue;

            }

            names[size++] = to\_string(id) + " " + name;

            cout << "ID " << id << " assigned to the name.\n";

            system("cls");

        }

    } else {

        cout << "Maximum number of names reached.\n";

    }

    head();

    cout << "\n\n\n\n\t\t\t\t\t\t  ADD RECORD\n\n\n\n";

    for (int i = 0; i < size; ++i) {

        int spaceIndex = names[i].find(" ");

        int id = stoi(names[i].substr(0, spaceIndex));

        string name = names[i].substr(spaceIndex + 1);

        cout << "\tID no. : " << id;

        cout << "\n\tName   : " << name << "\n\n";

    }

    cout << "\n\n\t    Press any key to continue...";

    getch();

    system("cls");

}

//case 2

void deleteNameById(string names[], int& size, int idToDelete) {

    while (true) {

        while (cin.fail()) {

            cin.clear();

            cin.ignore(numeric\_limits<streamsize>::max(), '\n');

            head();

            cout << "\n\n\n\n\t\t\t\t\t\t  DELETE RECORD\n";

            cout << "\n\n\n\tRECORD NOT FOUND!";

            cout <<"\n\n\tEnter ID again: ";

            cin >> idToDelete;

            system("cls");

        }

        bool found = false;

        for (int i = 0; i < size; ++i) {

            int id = stoi(names[i].substr(0, names[i].find(" ")));

            if (id == idToDelete) {

                head();

                cout << "\n\n\n\n\t\t\t\t\t\t  DELETE RECORD\n";

                cout << "\n\n\t\tID no. : " << id;

                size\_t spaceIndex = names[i].find(" ");

                string name = names[i].substr(spaceIndex + 1);

                cout << "\n\t\tName   : " << name << "\n\n";

                cout << "\n\t\tAre you sure you want to delete this entry? [Y/y] or [N/n]: ";

                char confirm;

                cin >> confirm;

                system("cls");

                if (confirm == 'Y' || confirm == 'y') {

                    head();

                    cout << "\n\n\n\n\t\t\t\t\t\t  DELETE RECORD\n";

                    for (int j = i; j < size - 1; ++j) {

                        names[j] = names[j + 1];

                    }

                    --size;

                    cout << "\n\n\t\tID " << idToDelete << " deleted.\n";

                    cout << "\n\n\t\tPress any key to continue...";

                    getch();

                    system("cls");

                } else {

                    head();

                    cout << "\n\n\n\n\t\t\t\t\t\t  DELETE RECORD\n";

                    cout << "\n\n\t\tDeletion canceled.\n";

                    cout << "\n\n\t\tPress any key to continue...";

                    getch();

                    system("cls");

                }

                return;

            }

        }

        if (!found) {

            head();

            cout << "\n\n\n\n\t\t\t\t\t\t  DELETE RECORD\n";

            cout << "\n\n\n\tRECORD NOT FOUND!";

            cout <<"\n\n\tEnter ID again: ";

            cin >> idToDelete;

            system("cls");

        }

    }

}

// case 3

void searchNameById(string names[], int size, int idToSearch) {

    bool recordFound = false;

    do {

        while (cin.fail()) {

            cin.clear();

            cin.ignore(numeric\_limits<streamsize>::max(), '\n');

            head();

            cout << "\n\n\n\n\t\t\t\t\t\t  SEARCH RECORD\n";

            cout << "\n\n\n\tRECORD NOT FOUND!";

            cout <<"\n\n\tEnter ID again: ";

            cin >> idToSearch;

            system("cls");

        }

        for (int i = 0; i < size; ++i) {

            int id = stoi(names[i].substr(0, names[i].find(" ")));

            if (id == idToSearch) {

                head();

                cout << "\n\n\n\n\t\t\t\t\t\t  SEARCH RECORD\n";

                cout << "\n\n\n\tRecord found with a registered ID & NAME: ";

                int spaceIndex = names[i].find(" ");

                int id = stoi(names[i].substr(0, spaceIndex));

                string name = names[i].substr(spaceIndex + 1);

                cout << "\n\n\t\tID no. : " << id;

                cout << "\n\t\tName   : " << name << "\n\n";

                cout << "\n\n\t    Press any key to continue...";

                getch();

                system("cls");

                recordFound = true;

                break;

            }

        }

        if (!recordFound) {

            head();

            cout << "\n\n\n\n\t\t\t\t\t\t  SEARCH RECORD\n";

            cout << "\n\n\n\tRECORD NOT FOUND!";

            cout <<"\n\n\tEnter ID again: ";

            cin >> idToSearch;

            system("cls");

        }

    } while (!recordFound);

}

//case 4

void modi(string names[], int size, int idModi) {

    while (true) {

        while (cin.fail()) {

            cin.clear();

            cin.ignore(numeric\_limits<streamsize>::max(), '\n');

            head();

            cout << "\n\n\n\n\t\t\t\t\t\t  SEARCH RECORD\n";

            cout << "\n\n\n\tRECORD NOT FOUND!";

            cout <<"\n\n\tEnter ID again: ";

            cin >> idModi;

            system("cls");

        }

        for (int i = 0; i < size; ++i) {

            int id = stoi(names[i].substr(0, names[i].find(" ")));

            if (id == idModi) {

                head();

                cout << "\n\n\n\n\t\t\t\t\t\t  SEARCH RECORD\n";

                cout << "\n\n\t\tID no. : " << id;

                size\_t spaceIndex = names[i].find(" ");

                string name = names[i].substr(spaceIndex + 1);

                cout << "\n\t\tName   : " << name << "\n\n";

                cout << "\n\t\tEnter the new name: ";

                string newNam;

                cin.ignore();

                getline(cin, newNam);

                system("cls");

                head();

                cout << "\n\n\n\n\t\t\t\t\t\t  MODIFY RECORD\n";

                names[i] = to\_string(id) + " " + newNam;

                cout << "\n\n\t\tID No   : " << idModi ;

                cout << "\n\t\tNew Name: " << newNam << "\n";

                cout << "\n\n\t\t   Press any key to continue...";

                getch();

                system("cls");

                return;

            }

        }

        head();

        cout << "\n\n\n\n\t\t\t\t\t\t  DISPLAY RECORD\n";

        cout << "\n\n\n\tRECORD NOT FOUND!";

        cout <<"\n\n\tEnter ID again: ";

        cin >> idModi;

        system("cls");

    }

}

// case 5

void disNames(string names[], int size) {

    head();

    cout << "\n\n\n\n\t\t\t\t\t\t  DISPLAY RECORD\n";

    cout << "\n\n\tList of Students:\n\n";

    for (int i = 0; i < size; ++i) {

        int spaceIndex = names[i].find(" ");

        int id = stoi(names[i].substr(0, spaceIndex));

        string name = names[i].substr(spaceIndex + 1);

        cout << "\t\tID no. : " << id;

        cout << "\n\t\tName   : " << name << "\n\n";

    }

    cout << "\n\tPress any key to continue...";

    getch();

    system("cls");

}

//case 6

void updatePassword() {

    while (true) {

        char currentPassword[10];

        head();

        cout << "\n\n\n\n\t\t\t\t\t\t  UPDATE PASSWORD\n";

        cout << "\n\n\n\tEnter the current password: ";

        getPassword(currentPassword);

        if (strcmp(currentPassword, globalPassword) == 0) {

            system("cls");

            break;

        } else {

            cout << "\n\n\tIncorrect current password. Please try again.\n";

            cout << "\n\tPress any key to continue...";

            getch();

            system("cls");

        }

    }

    char newPassword[10];

    while (true) {

        head();

        cout << "\n\n\n\n\t\t\t\t\t\t  UPDATE PASSWORD\n";

        cout << "\n\n\tEnter the new password: ";

        getPassword(newPassword);

        char confirmNewPassword[10];

        cout << "\n\tConfirm the new password: ";

        getPassword(confirmNewPassword);

        system("cls");

        if (strcmp(newPassword, confirmNewPassword) == 0) {

            break;

        } else {

            head();

            cout << "\n\n\n\n\t\t\t\t\t\t  UPDATE PASSWORD\n";

            cout << "\n\n\tNew password and confirmation do not match. Please try again.\n";

            cout << "\n\tPress any key to continue...";

            getch();

            system("cls");

        }

    }

    strcpy(globalPassword, newPassword); // maoy mag update sa global password

    head();

    cout << "\n\n\n\n\t\t\t\t\t\t  UPDATE PASSWORD\n";

    cout << "\n\n\tPassword updated successfully!\n";

    cout << "\n\tPlease Sign In Again for Confimation\n";

    cout << "\n\n\tPress any key to SIGN IN again...";

    getch();

    system("cls");

}

int main() {

string names[MAX\_NAMES];

int size = 0;

while (true) {

    login:

    char option[3], username[10], password[10];

    int firstinput = 1, login = 1;

        passHead();

        printf("\n\n\n\n\t\t\t\t\t     Username: ");

        scanf("%s", username);

        printf("\t\t\t\t\t     Password: ");

        getPassword(password);

        system("cls");

        if (strcmp(username, "admin") == 0 && strcmp(password, globalPassword) == 0) {

            cout << "\n\n\n\n\n\n\t\t\t\t\t    AUTHENTICATION COMPLETE";

            cout << "\n\t\t\t\t\t     Welcome to the System!";

            cout << "\n\n\t\t\t\t          Press any key to continue...";

            \_getch();

            system("cls");

        } else {

            cout << "\n\n\n\n\n\n\t\t\t\t\t    AUTHENTICATION FAILED";

            cout << "\n\t\t\t\t [Wrong user or password and please try again]";

            cout << "\n\n\n\n\t\t\t\t          Press any key to continue...";

            \_getch();

            system("cls");

            goto login;

        }

    here:

    head();

    cout << "\n\n\n\n\t\t\t\t\t\t  MAIN MENU\n";

    cout << "\n\n\n\tOptions to Perform:\n";

    cout << "\t\tPress [1] ADD Record\n";

    cout << "\t\tPress [2] DELETE Record\n";

    cout << "\t\tPress [3] SEARCH Record\n";

    cout << "\t\tPress [4] MODIFY Record\n";

    cout << "\t\tPress [5] DISPLAY Record\n";

    cout << "\t\tPress [6] UPDATE PASSWORD\n";

    cout << "\t\tPress [7] Exit\n\n";

    cout << "\tSelect Option: ";

    int against = 1;

    int choice;

    cin >> choice;

    switch (choice) {

        case 1:

            system("cls");

            getNameWithId(names, size);

            goto here;

        case 2:

            system("cls");

            head();

            cout << "\n\n\n\n\t\t\t\t\t\t  DELETE RECORD\n";

            cout << "\n\n\n\tEnter the ID: ";

            int idToDelete;

            cin >> idToDelete;

            system("cls");

            deleteNameById(names, size, idToDelete);

            goto here;

        case 3:

            system("cls");

            head();

            cout << "\n\n\n\n\t\t\t\t\t\t  SEARCH RECORD\n";

            cout << "\n\n\n\tEnter the ID: ";

            int idToSearch;

            cin >> idToSearch;

            system("cls");

            searchNameById(names, size, idToSearch);

            goto here;

        case 4:

            system("cls");

            head();

            cout << "\n\n\n\n\t\t\t\t\t\t  MODIFY RECORD\n";

            cout << "\n\n\n\tEnter the ID to modify: ";

            int idModi;

            cin >> idModi;

            system("cls");

            modi(names, size, idModi);

            goto here;

        case 5:

            system("cls");

            disNames(names, size);

            goto here;

        case 6:

            system("cls");

            updatePassword();

            break;

        case 7:

            system("cls");

                while (against) {

                head();

                cout << "\n\n\n\n\t\t\t\t\t\t  EXIT PROGRAM\n";

                cout << "\n\n\n\tOptions to Perform:\n";

                cout << "\t\tPress [Y/y] EXIT the Program\n";

                cout << "\t\tPress [N/n] RETURN to Main Menu\n\n";

                cout << "\tSelect Option: ";

                scanf("%s", option);

                if ((strcmp(option, "y") == 0 || strcmp(option, "Y") == 0 || strcmp(option, "n") == 0 || strcmp(option, "N") == 0)) {

                    if (strcmp(option, "y") == 0 || strcmp(option, "Y") == 0) {

                        system("cls");

                        break;

                    }

                    else {

                        system("cls");

                        goto here;

                        break;

                    }

                }

                else {

                    system("cls");

                    head();

                    cout << "\n\n\n\n\t\t\t\t\t\t  EXIT PROGRAM\n";

                    printf("\n\nWrong Key!");

                    printf("\nPlease type only [Y/y] for YES and [N/n] for NO.");

                    cout << "Press any key to continue...";

                    getch();

                    system("cls");

                }

            }

            if (strcmp(option, "n") == 0 || strcmp(option, "N") == 0) {

                break;

            }

        break;

        default:

        system("cls");

            head();

            cout << "\n\n\n\n\t\t\t\t\t\t  WARNING!!\n";

            cout << "\n\n\n\tInvalid option! Please select a valid option.\n";

            cout << "\n\tPress any key to continue...";

            getch();

            cin.clear();

            cin.ignore(numeric\_limits<streamsize>::max(), '\n');

            system("cls");

            goto here;

        }

    }

return 0;

}