```
#include <iostream>

#include <string>

#include <queue>

#include <map>

#include <ctime>


using namespace std;


// --- STATUS ENUM  ---

enum Status { PENDING, SHORTLISTED, REJECTED };


// --- STRUCT DEFINITION ---

struct Application {

    int id;

    string name;

    string email;

    int qualificationScore;

    Status status;

    time_t timestamp;

    Application* next;

};


// --- GLOBAL VARIABLES ---

Application* head = NULL;                // Linked list head

map<int, Application*> appHash;          // Hash (map) table for quick search

queue<int> interviewQueue;               // FIFO queue


// --- UTILITY FUNCTIONS ---

string statusToString(Status s) {

    if (s == PENDING) return "PENDING";

    else if (s == SHORTLISTED) return "SHORTLISTED";
    else if (s == REJECTED) return "REJECTED";
    return "UNKNOWN";
}

void printApplication(Application* a) {
    if (a == NULL) return;
    char* timeStr = ctime(&a->timestamp); // convert time to readable string

    cout << "ID: " << a->id
         << " | Name: " << a->name
         << " | Email: " << a->email
         << " | Score: " << a->qualificationScore
         << " | Status: " << statusToString(a->status)
         << " | Registered: " << timeStr;
}

// --- 1) ADD APPLICATION ---
void addApplication(int id, string name, string email, int score) {
    if (appHash.find(id) != appHash.end()) {
        cout << "Error: Application with ID " << id << " already exists.\n";
        return;
    }

    Application* node = new Application;
    node->id = id;
    node->name = name;
    node->email = email;
    node->qualificationScore = score;
    node->status = PENDING;
    node->timestamp = time(0);
    node->next = NULL;
    if (head == NULL) head = node;
    else {
        Application* cur = head;
        while (cur->next != NULL) cur = cur->next;
        cur->next = node;
    }

    appHash[id] = node;
    cout << "Successfully registered! Application ID: " << id << endl;
}

// --- 2) SCHEDULE INTERVIEW ---
void scheduleInterview(int id) {
    if (appHash.find(id) == appHash.end()) {
        cout << "No application with ID " << id << endl;
        return;
    }
    interviewQueue.push(id);
    cout << "Application " << id << " added to interview queue.\n";
}

// --- 3) SHOW INTERVIEW QUEUE ---
void showInterviewQueue() {
    if (interviewQueue.empty()) {
        cout << "Interview queue is empty.\n";
        return;
    }

    queue<int> temp = interviewQueue;
    cout << "Interview queue (front  back): ";
    while (!temp.empty()) {
        int id = temp.front(); temp.pop();
        cout << id << " ";
    }
    cout << endl;
}

// --- 4) PROCESS NEXT INTERVIEW ---
void processNextInterview() {
    if (interviewQueue.empty()) {
        cout << "No interviews scheduled.\n";
        return;
    }
    int id = interviewQueue.front();
    interviewQueue.pop();
    cout << "Processing interview for:\n";
    printApplication(appHash[id]);
}

// --- 5) SEARCH APPLICATION BY ID ---
void searchApplicationByID(int id) {
    if (appHash.find(id) == appHash.end()) {
        cout << "Application ID " << id << " not found.\n";
        return;
    }
    printApplication(appHash[id]);
}

// --- 6) RANK CANDIDATES ---
void showRankedCandidates() {
    if (appHash.empty()) {
        cout << "No candidates registered.\n";
        return;
    }

    // Convert map to vector for sorting
    vector<Application*> v;
    for (map<int, Application*>::iterator it = appHash.begin(); it != appHash.end(); ++it)
        v.push_back(it->second);

    // Simple bubble sort by qualificationScore (descending)
    for (int i = 0; i < v.size() - 1; i++) {
        for (int j = 0; j < v.size() - i - 1; j++) {
            if (v[j]->qualificationScore < v[j + 1]->qualificationScore)
                swap(v[j], v[j + 1]);
        }
    }

    cout << "Ranked Candidates:\n";
    for (int i = 0; i < v.size(); i++) {
        cout << i + 1 << ". ID: " << v[i]->id
             << " | Name: " << v[i]->name
             << " | Score: " << v[i]->qualificationScore
             << " | Status: " << statusToString(v[i]->status) << endl;
    }
}

// --- 7) UPDATE APPLICATION STATUS ---
void updateApplicationStatus(int id, int s) {
    if (appHash.find(id) == appHash.end()) {
        cout << "Application ID not found.\n";
        return;
    }
    if (s < 0 || s > 2) {
        cout << "Invalid status.\n";
        return;
    }
    appHash[id]->status = (Status)s;
    cout << "Updated status for ID " << id << "  " << statusToString((Status)s) << endl;
}

// --- 8) SHOW ALL APPLICATIONS ---
void showAllApplications() {
    if (head == NULL) {
        cout << "No applications registered yet.\n";
        return;
    }
    Application* cur = head;
    cout << "All Registered Applications:\n";
    while (cur != NULL) {
        printApplication(cur);
        cur = cur->next;
    }
}

// --- MENU ---
void showMenu() {
    cout << "\n===== JOB APPLICATION PORTAL =====\n";
    cout << "1. Register new application\n";
    cout << "2. Schedule interview\n";
    cout << "3. Show interview queue\n";
    cout << "4. Process next interview\n";
    cout << "5. Search application by ID\n";
    cout << "6. Show ranked candidates\n";
    cout << "7. Update application status\n";
    cout << "8. Show all applications\n";
    cout << "0. Exit\n";
    cout << "Choose option: ";
}

// --- MAIN ---
int main() {
    int choice;
    cout << "Welcome to the Job Application Portal!\n";

    while (true) {
        showMenu();
        cin >> choice;

        if (choice == 0) {
            cout << "Exiting...\n";
            break;
        } else if (choice == 1) {
            int id, score;
            string name, email;
            cout << "Enter ID: "; cin >> id;
            cin.ignore();
            cout << "Enter name: "; getline(cin, name);
            cout << "Enter email: "; getline(cin, email);
            cout << "Enter qualification score: "; cin >> score;
            addApplication(id, name, email, score);
        } else if (choice == 2) {
            int id; cout << "Enter ID: "; cin >> id;
            scheduleInterview(id);
        } else if (choice == 3) {
            showInterviewQueue();
        } else if (choice == 4) {
            processNextInterview();
        } else if (choice == 5) {
            int id; cout << "Enter ID: "; cin >> id;
            searchApplicationByID(id);
        } else if (choice == 6) {
            showRankedCandidates();
        } else if (choice == 7) {
            int id, s;
            cout << "Enter ID: "; cin >> id;
            cout << "Enter new status (0=PENDING, 1=SHORTLISTED, 2=REJECTED): ";
            cin >> s;
            updateApplicationStatus(id, s);
        } else if (choice == 8) {
            showAllApplications();
        } else {
            cout << "Invalid choice.\n";
        }
    }

    return 0;
}
```