ASSIGNMENT 04

Code:

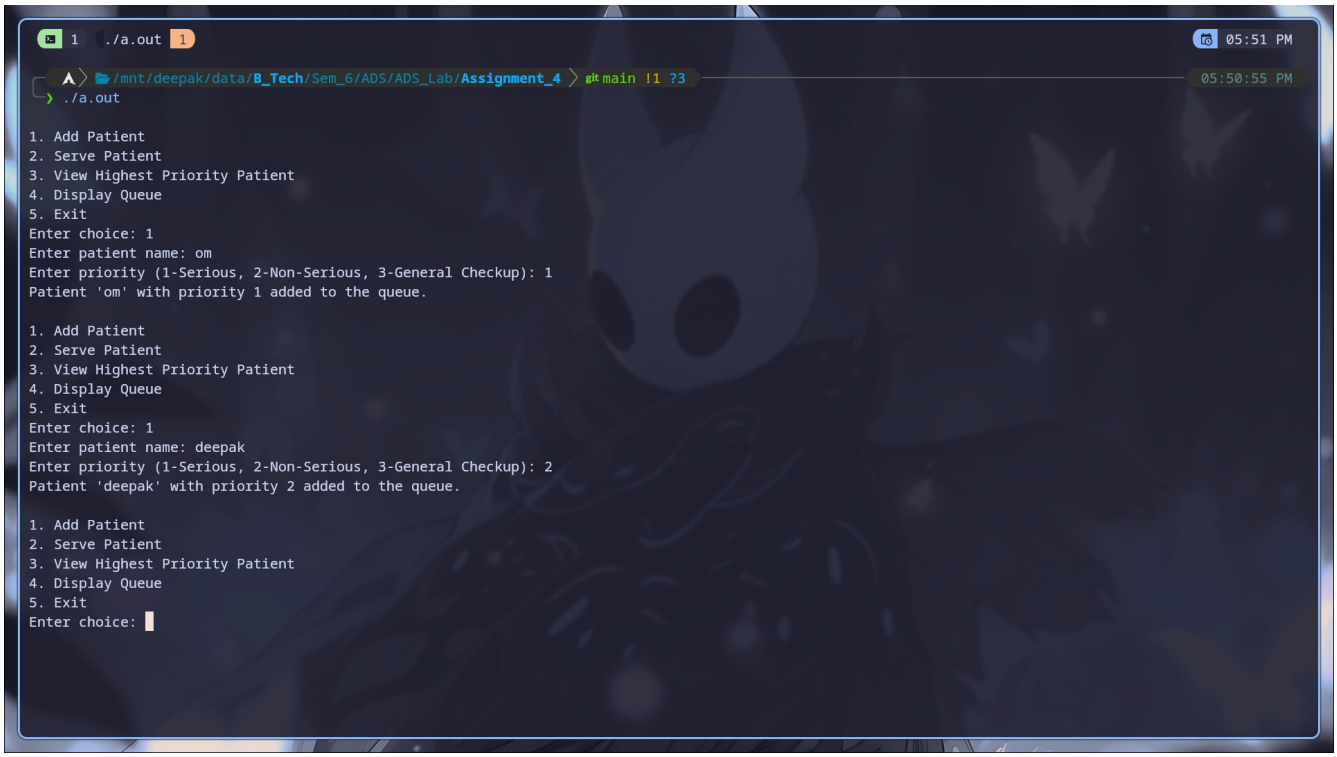
```
#include <iostream>
#include <queue>
#include <vector>

using namespace std;

// Struct to represent a patient with a name and priority level
struct Patient {
  string name;
  int priority;

  // Operator overload to compare patients based on priority (Min-Heap
behavior)
  bool operator>(const Patient &other) const {
    return priority > other.priority;
  }
};

class HospitalQueue {
private:
  // Min-Heap priority queue to store patients (higher priority = lower
number)
  priority_queue<Patient, vector<Patient>, greater<Patient>> pq;

public:
  // Function to insert a patient into the queue
  void insertPatient(string name, int priority) {
    pq.push({name, priority});
    cout << "Patient '" << name << "' with priority " << priority
         << " added to the queue.\n";
  }
```

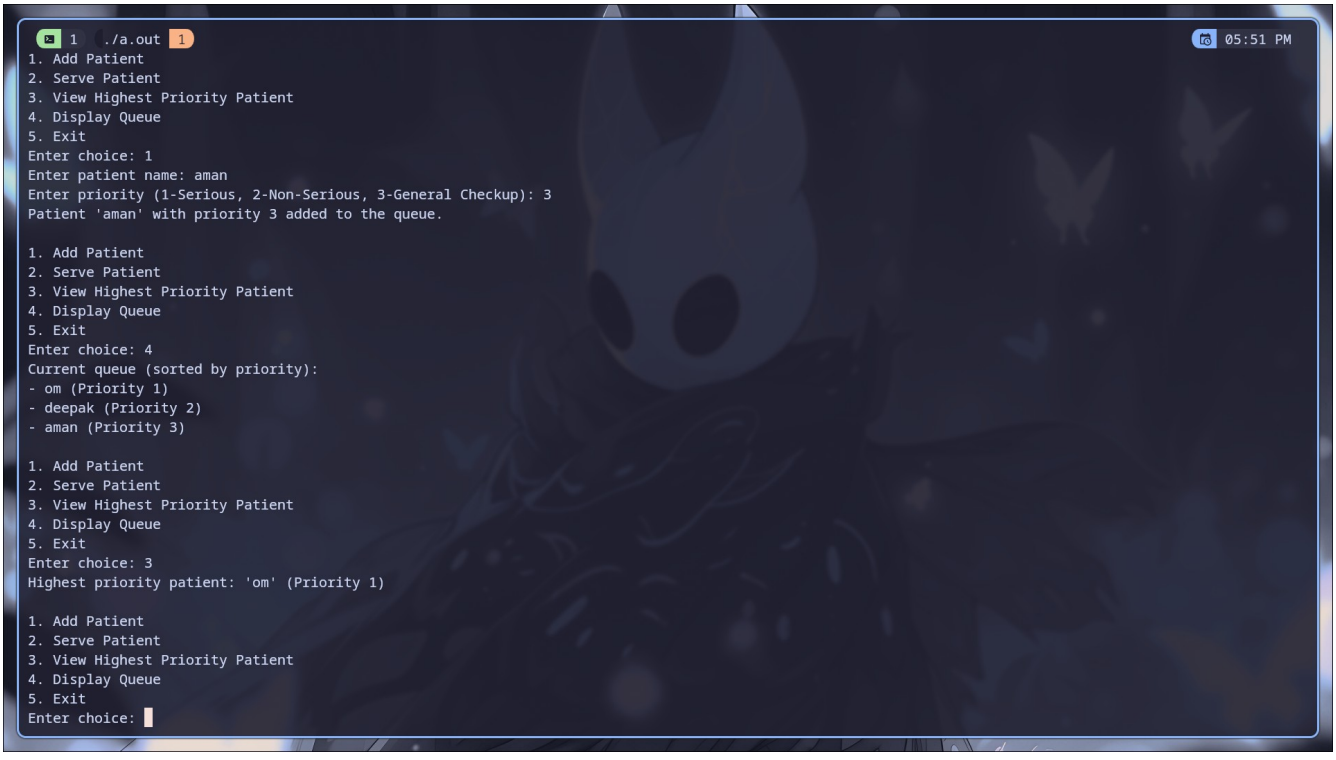
```
  // Function to serve the highest-priority patient
  void servePatient() {
    if (!pq.empty()) {
      Patient p = pq.top(); // Get the patient with the highest priority
      pq.pop(); // Remove the served patient
      cout << "Patient '" << p.name << "' (Priority " << p.priority
           << ") has been served.\n";
    } else {
      cout << "No patients in queue.\n";
    }
  }

  // Function to view the highest-priority patient without removing them
  void highestPriorityPatient() {
    if (!pq.empty()) {
      Patient p = pq.top();
      cout << "Highest priority patient: '" << p.name << "' (Priority "
           << p.priority << ")\n";
    } else {
      cout << "No patients in queue.\n";
    }
  }

  // Function to display all patients in the queue in order of priority
  void displayQueue() {
    if (pq.empty()) {
      cout << "No patients in queue.\n";
      return;
    }

    // Create a temporary copy of the queue to display without modifying
original
    priority_queue<Patient, vector<Patient>, greater<Patient>> tempPQ = pq;

    cout << "Current queue (sorted by priority):\n";
```

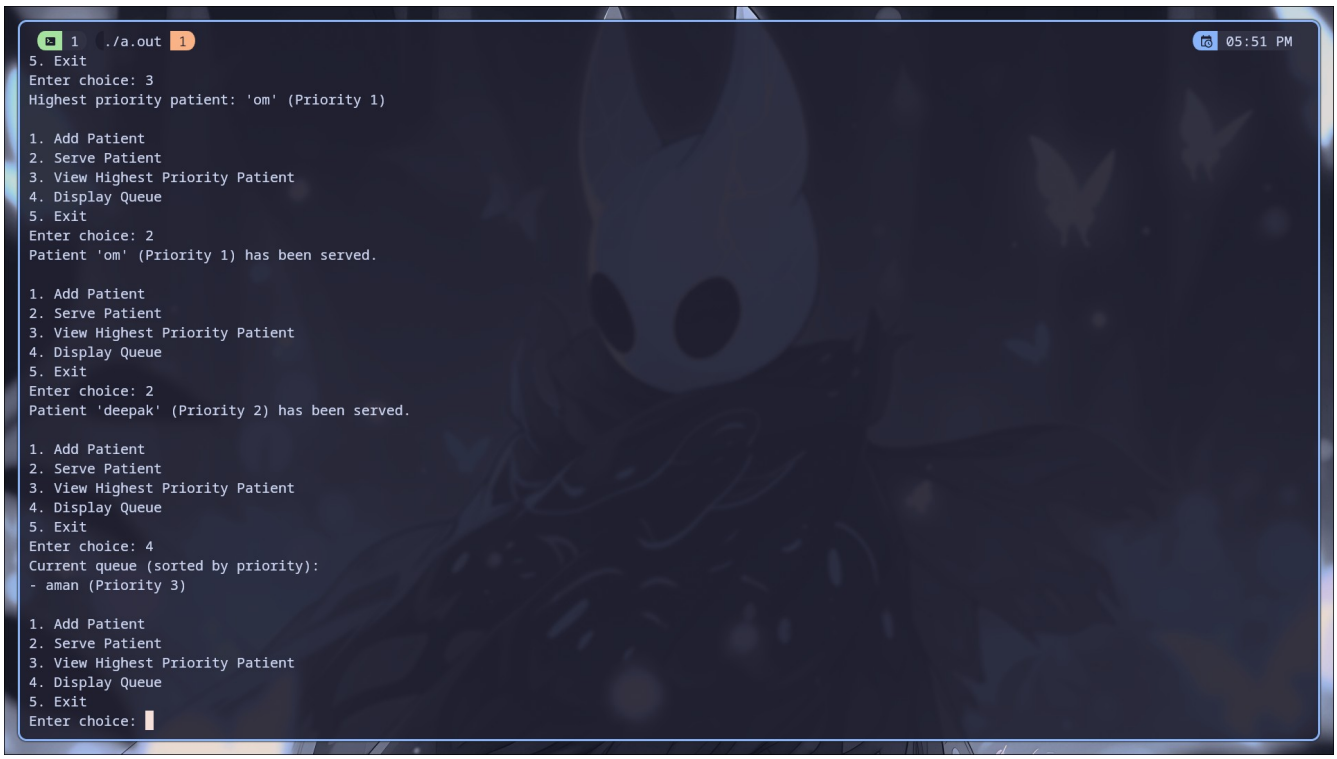
```
    while (!tempPQ.empty()) {
      Patient p = tempPQ.top();
      tempPQ.pop();
      cout << "- " << p.name << " (Priority " << p.priority << ")\n";
    }
  }
};

int main() {
  HospitalQueue hospital;
  int choice;

  while (true) {
    // Display menu options
    cout << "\n1. Add Patient\n2. Serve Patient\n3. View Highest Priority "
          "Patient\n4. Display Queue\n5. Exit\nEnter choice: ";
    cin >> choice;

    if (choice == 1) {
      // Get patient details from user
      string name;
      int priority;
      cout << "Enter patient name: ";
      cin >> name;
      cout << "Enter priority (1-Serious, 2-Non-Serious, 3-General Checkup):
";
      cin >> priority;
      hospital.insertPatient(name, priority);
    } else if (choice == 2) {
      hospital.servePatient();
    } else if (choice == 3) {
      hospital.highestPriorityPatient();
    } else if (choice == 4) {
      hospital.displayQueue();
    } else if (choice == 5) {
      break; // Exit the program
```

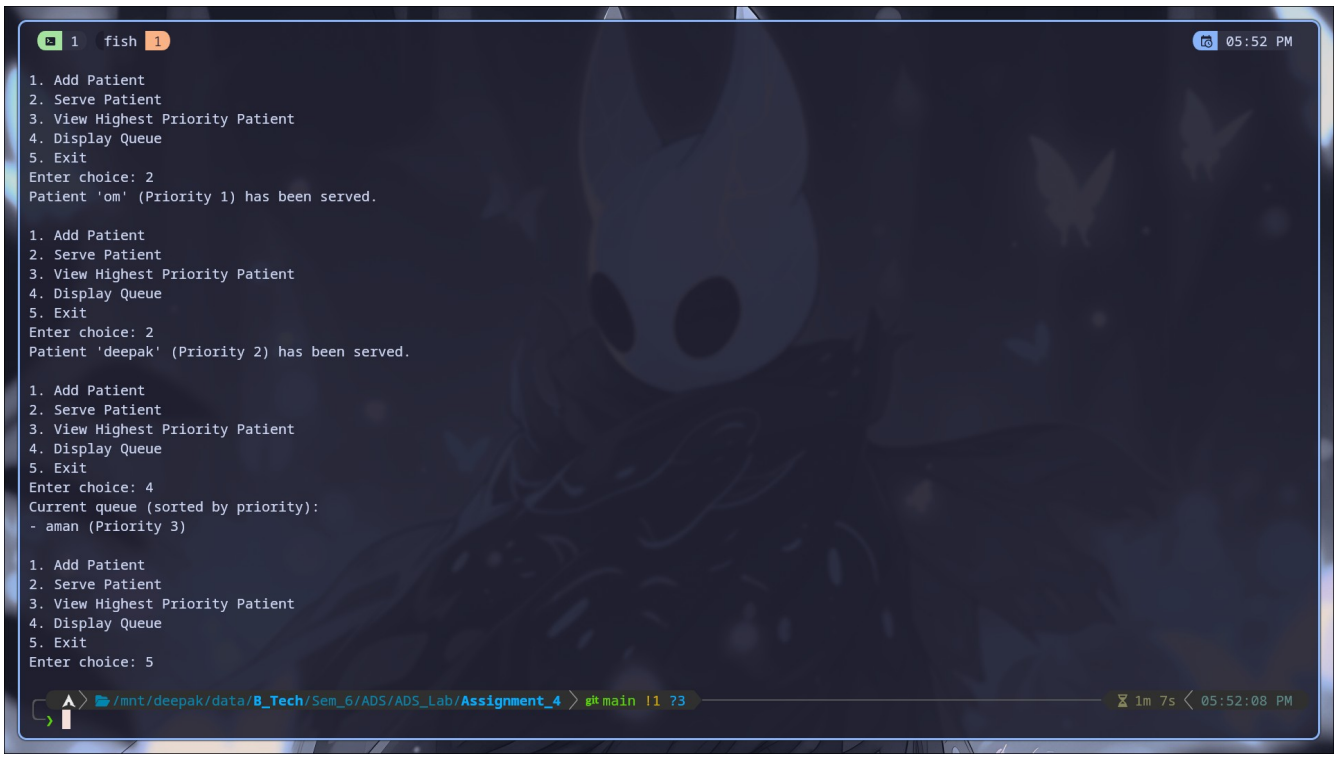
```
    } else {
      cout << "Invalid choice. Try again.\n";
    }
  }

  return 0;
}
```

Output:


1 ./a.out 1
05:51 PM
/mnt/deepak/data/B_Tech/Sem_6/ADS/ADS_Lab/Assignment_4 git main !1 ?3
05:50:55 PM
./a.out
1. Add Patient
2. Serve Patient
3. View Highest Priority Patient
4. Display Queue
5. Exit
Enter choice: 1
Enter patient name: om
Enter priority (1-Serious, 2-Non-Serious, 3-General Checkup): 1
Patient 'om' with priority 1 added to the queue.
1. Add Patient
2. Serve Patient
3. View Highest Priority Patient
4. Display Queue
5. Exit
Enter choice: 1
Enter patient name: deepak
Enter priority (1-Serious, 2-Non-Serious, 3-General Checkup): 2
Patient 'deepak' with priority 2 added to the queue.
1. Add Patient
2. Serve Patient
3. View Highest Priority Patient
4. Display Queue
5. Exit
Enter choice:


1 ./a.out 1
05:51 PM
1. Add Patient
2. Serve Patient
3. View Highest Priority Patient
4. Display Queue
5. Exit
Enter choice: 1
Enter patient name: aman
Enter priority (1-Serious, 2-Non-Serious, 3-General Checkup): 3
Patient 'aman' with priority 3 added to the queue.
1. Add Patient
2. Serve Patient
3. View Highest Priority Patient
4. Display Queue
5. Exit
Enter choice: 4
Current queue (sorted by priority):
- om (Priority 1)
- deepak (Priority 2)
- aman (Priority 3)
1. Add Patient
2. Serve Patient
3. View Highest Priority Patient
4. Display Queue
5. Exit
Enter choice: 3
Highest priority patient: 'om' (Priority 1)
1. Add Patient
2. Serve Patient
3. View Highest Priority Patient
4. Display Queue
5. Exit
Enter choice:


1 ./a.out 1
05:51 PM
5. Exit
Enter choice: 3
Highest priority patient: 'om' (Priority 1)
1. Add Patient
2. Serve Patient
3. View Highest Priority Patient
4. Display Queue
5. Exit
Enter choice: 2
Patient 'om' (Priority 1) has been served.
1. Add Patient
2. Serve Patient
3. View Highest Priority Patient
4. Display Queue
5. Exit
Enter choice: 2
Patient 'deepak' (Priority 2) has been served.
1. Add Patient
2. Serve Patient
3. View Highest Priority Patient
4. Display Queue
5. Exit
Enter choice: 4
Current queue (sorted by priority):
- aman (Priority 3)
1. Add Patient
2. Serve Patient
3. View Highest Priority Patient
4. Display Queue
5. Exit
Enter choice:


1 fish 1
05:52 PM
1. Add Patient
2. Serve Patient
3. View Highest Priority Patient
4. Display Queue
5. Exit
Enter choice: 2
Patient 'om' (Priority 1) has been served.
1. Add Patient
2. Serve Patient
3. View Highest Priority Patient
4. Display Queue
5. Exit
Enter choice: 2
Patient 'deepak' (Priority 2) has been served.
1. Add Patient
2. Serve Patient
3. View Highest Priority Patient
4. Display Queue
5. Exit
Enter choice: 4
Current queue (sorted by priority):
- aman (Priority 3)
1. Add Patient
2. Serve Patient
3. View Highest Priority Patient
4. Display Queue
5. Exit
Enter choice: 5
/mnt/deepak/data/B_Tech/Sem_6/ADS/ADS_Lab/Assignment_4
git main !1 ?3
1m 7s
05:52:08 PM