'''

Write C++ program for storing appointment schedule for day. Appointments are booked

randomly using linked list. Set start and end time and min and max duration for visit slot.

Write functions fora) Display free slots

b) Book appointment

c) Cancel appointment (check validity, time bounds, availability)

d) Sort list based on time

e) Sort list based on time using pointer manipulation

'''

```
#include <iostream>
using namespace std;

struct Appointment {
   int startHour, startMin, endHour, endMin;
   Appointment* next;
};

Appointment* head = nullptr;

// Function to display the free slots
void displayFreeSlots() {
   Appointment* temp = head;
```

```
    cout << "Free Slots: " << endl;

    while (temp != nullptr) {

        cout << "Start Time: " << temp->startHour << ":" << temp->startMin << endl;

        cout << "End Time: " << temp->endHour << ":" << temp->endMin << endl;

        temp = temp->next;

    }

}


// Function to book an appointment

void bookAppointment(int startHour, int startMin, int endHour, int endMin) {

    Appointment* newAppointment = new Appointment;

    newAppointment->startHour = startHour;

    newAppointment->startMin = startMin;

    newAppointment->endHour = endHour;

    newAppointment->endMin = endMin;

    newAppointment->next = nullptr;


    if (head == nullptr) {

        head = newAppointment;

    }

    else {

        Appointment* temp = head;

        while (temp->next != nullptr) {

            temp = temp->next;

        }
        temp->next = newAppointment;
    }
}

// Function to cancel an appointment
void cancelAppointment(int startHour, int startMin) {
    if (head == nullptr) {
        cout << "No appointments booked." << endl;
    }
    else {
        if (head->startHour == startHour && head->startMin == startMin) {
            Appointment* temp = head;
            head = head->next;
            delete temp;
            cout << "Appointment canceled." << endl;
        }
        else {
            Appointment* temp = head;
            Appointment* prev = nullptr;
            while (temp != nullptr && !(temp->startHour == startHour && temp->startMin == startMin)) {
                prev = temp;
                temp = temp->next;
            }
            if (temp == nullptr) {
                cout << "Appointment not found." << endl;
            }
            else {
                prev->next = temp->next;
                delete temp;
                cout << "Appointment canceled." << endl;
            }
        }
    }
}

// Function to sort the list based on time
void sortList() {
    if (head == nullptr || head->next == nullptr) {
        return;
    }

    Appointment* prev = nullptr;
    Appointment* current = head;
    Appointment* temp = nullptr;

    bool isSorted = false;
    while (!isSorted) {
        isSorted = true;
        while (current->next != nullptr) {
            if (current->startHour > current->next->startHour || (current->startHour == current->next-
>startHour && current->startMin > current->next->startMin)) {
```

```
                if (prev == nullptr) {
                    head = current->next;
                    temp = head->next;
                    head->next = current;
                }
                else {
                    prev->next = current->next;
                    temp = current->next->next;
                    current->next->next = current;
                }
                current->next = temp;
                prev = nullptr;
                isSorted = false;
            }
            else {
                prev = current;
                current = current->next;
            }
        }
        prev = nullptr;
        current = head;
    }
}

int main() {
```

```
// Initial list

bookAppointment(9, 0, 9, 30);

bookAppointment(10, 0, 10, 30);

bookAppointment(11, 0, 11, 30);

bookAppointment(9, 30, 10, 0);


// Display free slots

displayFreeSlots();


// Book an appointment

bookAppointment(12, 0, 12, 30);


// Display free slots after booking

displayFreeSlots();


// Cancel an appointment

cancelAppointment(10, 0);


// Display free slots after cancellation

displayFreeSlots();


// Sort the list based on time

sortList();


// Display free slots after sorting
```

```
    displayFreeSlots();


    return 0;

}
```