Assignment 3

Mayur Zope SE Comp A 75

A book consists of chapters, chapters consist of sections and sections consist of subsections. Construct a tree and print the nodes. Find the time and space requirements of your method.

```
#include <iostream>
#include <vector>
#include <string>
#include <memory>  // For smart pointers
using namespace std;
// Class to represent a Subsection
class Subsection {
public:
    string name;
    Subsection(string name) {
        this->name = name;
    }
};

// Class to represent a Section
class Section {
public:
    string name;
    vector<unique_ptr<Subsection>> subsections;

    Section(string name) {
        this->name = name;
    }

    // Add a Subsection to the section
    void addSubsection(unique_ptr<Subsection> subsection) {
        subsections.push_back(move(subsection));
```

```
    }

    // Print subsections
    void printSubsections() const {
        for (const auto& subsection : subsections) {
            cout << "\t\tSubsection: " << subsection->name << endl;
        }
    }
};

// Class to represent a Chapter
class Chapter {
public:
    string name;
    vector<unique_ptr<Section>> sections;

    Chapter(string name) {
        this->name = name;
    }

    // Add a Section to the chapter
    void addSection(unique_ptr<Section> section) {
        sections.push_back(move(section));
    }

    // Print sections
    void printSections() const {
        for (const auto& section : sections) {
            cout << "\tSection: " << section->name << endl;
            section->printSubsections(); // Print subsections under this section
```

```
        }
    }
};

// Class to represent a Book
class Book {
public:
    string title;
    vector<unique_ptr<Chapter>> chapters;

    Book(string title) {
        this->title = title;
    }

    // Add a Chapter to the book
    void addChapter(unique_ptr<Chapter> chapter) {
        chapters.push_back(move(chapter));
    }

    // Print the book structure
    void printBook() const {
        cout << "Book: " << title << endl;
        for (const auto& chapter : chapters) {
            cout << "Chapter: " << chapter->name << endl;
            chapter->printSections(); // Print sections and subsections for each
chapter
        }
    }
};
```

```
int main() {
    string bookTitle, chapterTitle, sectionTitle, subsectionTitle;
    int numChapters, numSections, numSubsections;

    // Taking input for the Book
    cout << "Enter the title of the book: ";
    getline(cin, bookTitle);

    unique_ptr<Book> book = make_unique<Book>(bookTitle);

    // Taking input for Chapters
    cout << "Enter the number of chapters: ";
    cin >> numChapters;
    cin.ignore(); // To ignore the newline character after the number input

    for (int i = 0; i < numChapters; ++i) {
        cout << "Enter title for Chapter " << (i + 1) << ": ";
        getline(cin, chapterTitle);

        unique_ptr<Chapter> chapter = make_unique<Chapter>(chapterTitle);

        // Taking input for Sections in the Chapter
        cout << "Enter the number of sections in chapter " << (i + 1) << ": ";
        cin >> numSections;
        cin.ignore();

        for (int j = 0; j < numSections; ++j) {
            cout << "Enter title for Section " << (j + 1) << ": ";
            getline(cin, sectionTitle);
```

```
            unique_ptr<Section> section = make_unique<Section>(sectionTitle);

            // Taking input for Subsections in the Section
            cout << "Enter the number of subsections in section " << (j + 1) << ": ";
            cin >> numSubsections;
            cin.ignore();

            for (int k = 0; k < numSubsections; ++k) {
                cout << "Enter title for Subsection " << (k + 1) << ": ";
                getline(cin, subsectionTitle);

                unique_ptr<Subsection> subsection =
make_unique<Subsection>(subsectionTitle);
                section->addSubsection(move(subsection));
            }

            chapter->addSection(move(section));
        }

        book->addChapter(move(chapter));
    }

    // Print the book structure
    book->printBook();

    // No need to manually clean up memory, as it's done automatically by smart
pointers
    return 0;
}
```

// OUTPUT

```
cc@CC01: ~/Downloads
File  Edit  View  Search  Terminal  Help
cc@CC01:~/Downloads$ g++ dsl3.cpp
cc@CC01:~/Downloads$ ./a.out
Enter the title of the book: India
Enter the number of chapters: 2
Enter title for Chapter 1: Rule
Enter the number of sections in chapter 1: 1
Enter title for Section 1: Rise
Enter the number of subsections in section 1: 0
Enter title for Chapter 2: Rampage
Enter the number of sections in chapter 2: 0
Book: India
Chapter: Rule
        Section: Rise
Chapter: Rampage
cc@CC01:~/Downloads$
```