Sir Syed CASE Institute of Technology

DSA PROJECT:

SUBMITTED TO:

Sir Zeeshan Aslam

SUBMITTED BY:

Ubaid Ahmad (2410-0011)

Project Title

Student Support Program

Student Support Program Documentation

Introduction

The Student Support Program is a C++-based application designed to assist students in finding suitable universities and scholarships based on their academic profiles. By leveraging data structures such as arrays, linked lists, stacks, queues, and binary search trees (BST), the program provides an efficient and user-friendly portal for students to explore educational opportunities and manage scholarship applications.

Objectives

- **University Search**: Enable students to find universities matching their GPA, preferred location, and program of study.
- **Scholarship Finder**: Identify scholarships based on GPA and major, sorted by award amount.
- **Application Management**: Allow students to apply for scholarships and track their applications.
- **Recommendation System**: Provide personalized university recommendations based on the student's major.
- **History Tracking**: Maintain a history of recommendations for quick reference.
- **Efficient Data Management**: Utilize appropriate data structures to ensure fast and reliable operations.

Scope

The program targets students seeking higher education opportunities in universities worldwide. It includes a sample dataset of universities and scholarships but is designed to be scalable for larger datasets. The system supports basic CRUD (Create, Read, Update, Delete) operations for data management and provides a menu-driven interface for user interaction.

Significance

This project addresses the challenges students face in navigating the complex landscape of university admissions and scholarship opportunities. By automating the search and application process, it saves time, reduces errors, and empowers students to make informed decisions about their education.

Procedures

System Workflow

1. **Initialization**:

    - The program loads sample data for universities and scholarships into a BST and array, respectively.
    - Data structures (Stack for recommendation history, Queue for applications) are initialized.
2. **User Interaction**:
    - Users interact via a menu-driven interface with options to:
        - Find universities by GPA, location, and program.
        - Find scholarships by GPA and major.
        - Apply for scholarships by specifying the award amount.
        - View scholarship applications.
        - Get university recommendations based on major.
        - View the most recent recommendation.
        - Exit the program.
3. **Data Processing**:
    - **University Search**: Filters universities from the BST based on user criteria.
    - **Scholarship Search**: Matches scholarships from the array and sorts them by award amount.
    - **Recommendations**: Pushes matching universities to a stack for history tracking.
    - **Applications**: Enqueues scholarship applications for tracking.
4. **Output**:
    - Results are displayed in a clear format, including university/scholarship details or error messages if no matches are found.

Implementation Details

- **Main Function**: Orchestrates the program flow, including menu display and user input handling.
- **Data Loading**: The `loadSampleData` function populates the BST with university data and an array with scholarship data.
- **Search Functions**:
    - `findUniversities`: Traverses the BST to find universities matching GPA, location, and program.
    - `findScholarships`: Scans the scholarship array and sorts results by award amount.
- **Recommendation System**: The `getRecommendations` function matches universities by major and stores them in a stack.
- **Application Management**: The `enqueue` function in the Queue class adds scholarship applications, and `display` shows the application list.
- **History Tracking**: The `push` and `displayRecent` functions in the Stack class manage recommendation history.

Error Handling

- Checks for empty data structures (e.g., empty stack/queue).
- Validates user input for GPA and scholarship amounts.
- Displays appropriate messages when no matches are found.

Data Structures Used

The program employs multiple data structures to achieve its functionality efficiently:

1. **Arrays**:
    - **Purpose**: Store scholarships and temporary results for universities and scholarships.
    - **Implementation**: Fixed-size arrays (`MAX_SCHOLARSHIPS`, `MAX_UNIVERSITIES`) for simplicity.
    - **Usage**: Scholarships are stored in an array for quick access during searches. Results arrays hold filtered universities/scholarships for display.
    - **Complexity**: O(n) for searching, $O(n^2)$ for sorting scholarships by award.
2. **Linked Lists**:
    - **Purpose**: Form the basis for Stack and Queue implementations.
    - **Implementation**: Doubly linked lists for Stack (recommendation history) and Queue (scholarship applications).
    - **Usage**: Enable dynamic memory allocation and efficient insertion/deletion.
    - **Complexity**: O(1) for push/pop (Stack) and enqueue (Queue), O(n) for display.
3. **Stack**:
    - **Purpose**: Store recommendation history for quick access to the most recent recommendation.
    - **Implementation**: Doubly linked list-based stack with `push`, `pop`, and `displayRecent` operations.
    - **Usage**: Pushes universities recommended to the user, allowing retrieval of the latest recommendation.
    - **Complexity**: O(1) for push/pop, O(1) for displaying the most recent item.
4. **Queue**:
    - **Purpose**: Manage scholarship applications in the order they are submitted.
    - **Implementation**: Doubly linked list-based queue with `enqueue` and `display` operations.
    - **Usage**: Enqueues scholarship applications and displays them in order.
    - **Complexity**: O(1) for enqueue, O(n) for display.
5. **Binary Search Tree (BST)**:
    - **Purpose**: Store and retrieve universities sorted by ranking.
    - **Implementation**: BST with `insert`, `inorderTraversal`, and `getSorted` operations.
    - **Usage**: Organizes universities by ranking for efficient retrieval during searches and recommendations.
    - **Complexity**: O(h) for insertion (where h is tree height), O(n) for inorder traversal to retrieve sorted universities.

Target Audience

Primary Audience

- **High School Students**: Seeking universities and scholarships for undergraduate studies.
- **Undergraduate Students**: Exploring transfer options or scholarships for advanced studies.
- **International Students**: Looking for universities in specific locations (e.g., USA, Pakistan) with programs matching their interests.

Secondary Audience

- **Academic Advisors**: Assisting students in finding suitable educational opportunities.
- **Parents/Guardians**: Supporting their children in navigating university and scholarship options.
- **Educational Institutions**: Using the program as a tool to guide students toward partner universities or scholarships.

System Features

1. **University Search**:
    - Filters universities by GPA, location, and program.
    - Displays name, ranking, and tuition for matching universities.
2. **Scholarship Finder**:
    - Matches scholarships by GPA and major.
    - Sorts results by award amount (descending) for user convenience.
3. **Scholarship Application**:
    - Allows users to apply for scholarships by specifying the award amount.
    - Validates scholarship existence before enqueuing.
4. **Recommendation System**:
    - Suggests universities based on the user's major.
    - Stores recommendations in a stack for history tracking.
5. **History Tracking**:
    - Displays the most recent university recommendation.
6. **User-Friendly Interface**:
    - Menu-driven console interface for easy navigation.
    - Clear prompts and error messages to guide users.

Conclusion

The Student Support Program is a robust and functional application that demonstrates the effective use of data structures to solve a real-world problem. By providing students with tools to find universities, scholarships, and manage applications, it serves as a valuable educational resource. The project's modular design and clear documentation make it suitable for academic evaluation and future expansion. With potential enhancements like database integration and a GUI, the program could evolve into a widely-used tool for students worldwide.

THANK YOU