

# SIX WEEKS SUMMER TRANING REPORT

ON

**DATA STRUCTURE AND ALGORITHM**

Programme Name : B.Tech CSE (3rd Year) Under the Guidance of Pavani Bandi

School of Computer Science & Engineering Lovely Professional University, Phagwara

**DECLARATION** I hereby declare that I completed my four-week summer training at the Board Infinity platform from May 28, 2024, to June 29, 2024, under the guidance of Pavani Bandi. I worked with full dedication during these four weeks of training, and my learning outcomes fulfil the requirements for the award of the B.Tech. degree in Computer Science and Engineering from Lovely Professional University, Phagwara.”

Date – 28/08/2024

**ACKNOWLEDGEMENT**

I would like to express my gratitude to both my university and Board Infinity for providing me with the golden opportunity to participate in this wonderful summer training program focused on Data Structure and Algorithm using C++. Through this experience, I not only completed assignments and learned extensively but also discovered many new things.

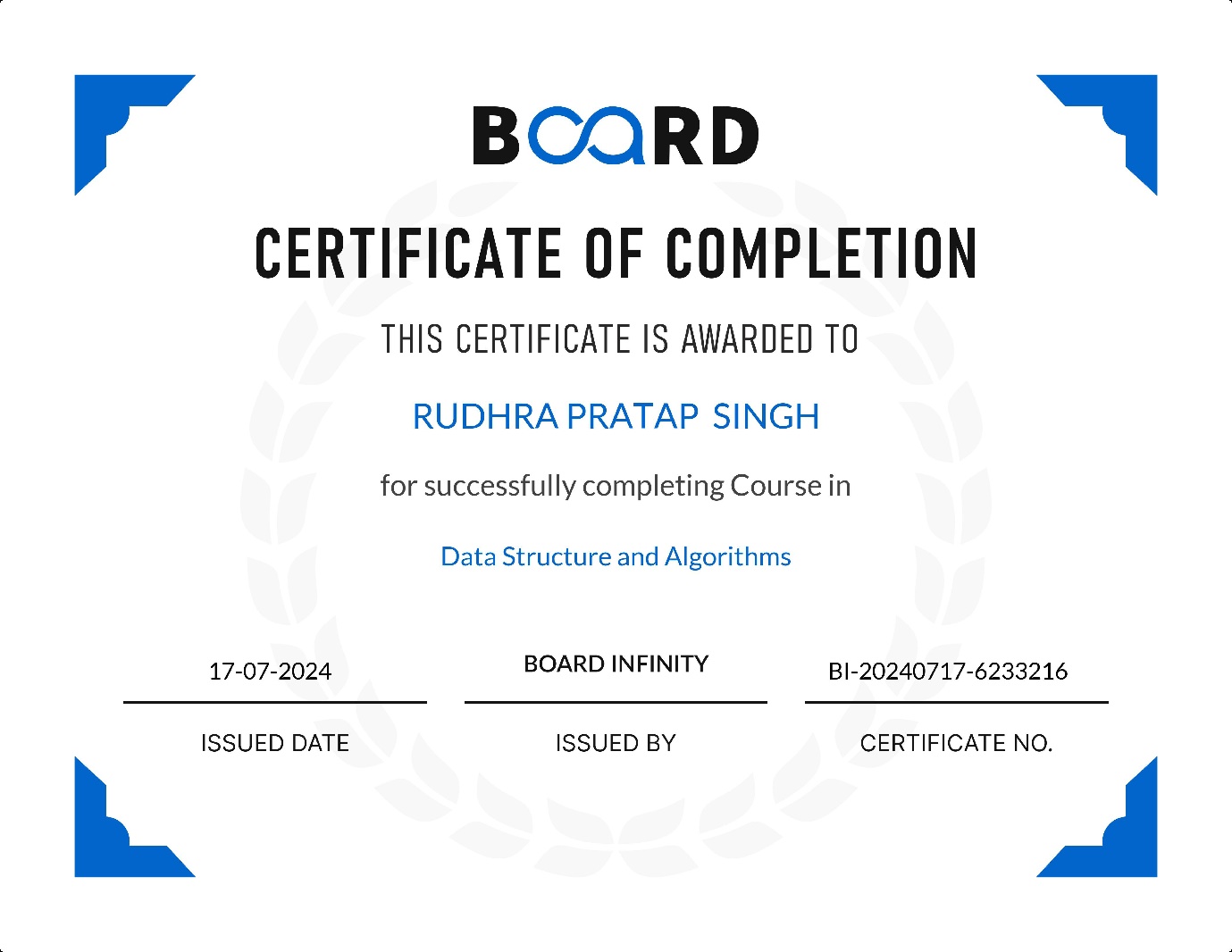
I am sincerely thankful to them.

Additionally, I extend my heartfelt appreciation to my friends who consistently supported me whenever I faced challenges related to my coursework. Their unwavering assistance has been invaluable, and I am fortunate to have such a strong support system.

While I put in significant effort into this project, it would not have been possible without the kind support and assistance of various individuals and organizations. I express my sincere thanks to all of them.

A special shout-out goes to our trainer, Pavani Bandi. His guidance, continuous encouragement, and meticulous review of our assignments have been instrumental. He took the time to thoroughly assess the project and training sessions, making necessary corrections whenever needed. We are immensely grateful for his dedication.

**Summer Training Certificate by Board Infinity**



**INDEX**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Title** | **Page No.** |
| 1 | Introduction | 6 |
| 2 | Technology Learnt | 7-9 |
| 3 | Reason for choosing R Programming | 10-11 |
| 4 | Project Overview and Analysis | 12-25 |
| 5 | Conclusion | 26 |
| 6 | Bibliography | 26 |

**INTRODUCTION**

**Overview**- Data Structures and Algorithms (DSA) are fundamental concepts in computer science that enable efficient data management and processing. Understanding DSA is crucial for developing efficient software applications and solving complex computational problems.

Data structures are specialized formats for organizing, processing, and storing data. They provide a way to manage large amounts of data efficiently. Common types of data structures include:

* Arrays: A collection of elements identified by index or key.
* Linked Lists: A linear collection of data elements, where each element points to the next.
* Stacks: A collection of elements that follows the Last In First Out (LIFO) principle.
* Queues: A collection of elements that follows the First In First Out (FIFO) principle.
* Trees: A hierarchical structure that consists of nodes, with a single node as the root.
* Graphs: A collection of nodes connected by edges, used to represent relationships.

Algorithms are step-by-step procedures or formulas for solving problems. They are essential for performing operations on data structures. Key characteristics of algorithms include:

* Efficiency: Measured in terms of time and space complexity.
* Correctness: The algorithm must produce the correct output for all valid inputs.
* Finiteness: An algorithm must terminate after a finite number of steps.

**Importance of DSA**

**Efficient Data Management**

Data structures provide a systematic way to organize and store data, enabling efficient access and modification. Choosing the right data structure can significantly enhance the performance of applications, especially when handling large datasets.

**Performance Optimization**

Algorithms are essential for performing operations on data structures. Efficient algorithms reduce time complexity, leading to faster execution and better resource utilization, which is vital in performance-critical applications.

**Problem Solving Skills**

Understanding DSA equips developers with the tools to approach complex problems systematically. It fosters analytical thinking and helps in devising effective solutions tailored to specific challenges.

**Scalability**

As applications grow, efficient data structures and algorithms ensure that they can scale effectively without a significant drop in performance, making them essential for modern software development.

**TECHNOLOGY LEARNT**

Data Structures and Algorithms (DSA) are the backbone of computer science and programming. They provide the necessary tools to efficiently manage, manipulate, and process data, which is crucial for solving complex problems and optimizing performance.

**Data Structures**

**1. Arrays**

* Definition: A collection of elements identified by index or key.
* Characteristics: Fixed size, contiguous memory allocation.
* Operations: Access (O(1)), Insertion (O(n)), Deletion (O(n)).
* Use Cases: Storing and accessing sequential data, implementing other data structures like heaps and hash tables.

**2. Linked Lists**

* Definition: A linear collection of nodes, where each node contains data and a reference to the next node.
* Types:
  + - Singly Linked List: Each node points to the next node.
    - Doubly Linked List: Each node points to both the next and previous nodes.
    - Circular Linked List: The last node points back to the first node.
* Operations: Insertion (O(1)), Deletion (O(1)), Access (O(n)).
* Use Cases: Dynamic memory allocation, implementing stacks and queues.

**3. Stacks**

* Definition: A collection of elements with LIFO (Last In, First Out) access.
* Operations: Push (O(1)), Pop (O(1)), Peek (O(1)).
* Use Cases: Function call management, expression evaluation, backtracking algorithms.

**4. Queues**

* Definition: A collection of elements with FIFO (First In, First Out) access.
* Types:
  + - Simple Queue: Basic FIFO structure.
    - Circular Queue: The last position is connected back to the first position.
    - Priority Queue: Elements are dequeued based on priority.
    - Deque (Double-Ended Queue): Insertion and deletion can occur at both ends.
* Operations: Enqueue (O(1)), Dequeue (O(1)), Peek (O(1)).
* Use Cases: Scheduling algorithms, buffering data streams.

**5. Trees**

* Definition: A hierarchical data structure consisting of nodes connected by edges.
* Types:
* Binary Tree: Each node has at most two children.
* Binary Search Tree (BST): A binary tree with ordered nodes.
* AVL Tree: A self-balancing binary search tree.
* Red-Black Tree: A balanced binary search tree with color properties.
* B-Tree: A self-balancing tree data structure that maintains sorted data.
* B+ Tree: An extension of B-Tree with improved search performance.
* Operations: Insertion (O(log n)), Deletion (O(log n)), Search (O(log n)).
* Use Cases: Database indexing, file systems, network routing.

**6.** Heaps

* Definition: A specialized tree-based data structure that satisfies the heap property.
* Types:
* Min-Heap: The parent node is less than or equal to its children.
* Max-Heap: The parent node is greater than or equal to its children.
* Operations: Insertion (O(log n)), Deletion (O(log n)), Peek (O(1)).
* Use Cases: Priority queues, heap sort, graph algorithms.

**7. Hash Tables**

* Definition: A data structure that maps keys to values using a hash function.
* Characteristics: Efficient for search, insertion, and deletion.
* Operations: Insertion (O(1)), Deletion (O(1)), Search (O(1)).
* Use Cases: Implementing associative arrays, database indexing.

**8. Graphs**

* Definition: A collection of vertices (nodes) and edges (connections).
* Types:

Directed Graph: Edges have a direction.

Undirected Graph: Edges do not have a direction.

Weighted Graph: Edges have weights.

Unweighted Graph: Edges do not have weights.

* Operations: Traversal (O(V + E)), Shortest Path (O(V^2) or O(E + V log V)).
* Use Cases: Social networks, transportation networks, network topology.

**Algorithms**

**1. Sorting Algorithms**

* Bubble Sort: Simple comparison-based sort (O(n^2)).
* Selection Sort: Selects the smallest element and swaps it (O(n^2)).
* Insertion Sort: Builds the sorted array one item at a time (O(n^2)).
* Merge Sort: Divide and conquer algorithm (O(n log n)).
* Quick Sort: Divide and conquer algorithm with pivot (O(n log n)).
* Heap Sort: Uses a heap data structure (O(n log n)).
* Radix Sort: Non-comparative integer sorting algorithm (O(nk)).

**2. Searching Algorithms**

* Linear Search: Sequentially checks each element (O(n)).
* Binary Search: Divides the search interval in half (O(log n)).

**3. Graph Algorithms**

* Depth-First Search (DFS): Explores as far as possible along each branch (O(V + E)).
* Breadth-First Search (BFS): Explores all neighbors at the present depth (O(V + E)).
* Dijkstra’s Algorithm: Finds the shortest path from a single source (O(V^2) or O(E + V log V)).
* Bellman-Ford Algorithm: Finds shortest paths from a single source in a weighted graph (O(VE)).
* Kruskal’s Algorithm: Finds the minimum spanning tree (O(E log E)).
* Prim’s Algorithm: Finds the minimum spanning tree (O(V^2) or O(E + V log V)).

**4. Dynamic Programming**

* Definition: Solves problems by breaking them down into simpler subproblems and storing the results.
* Examples:
* Fibonacci Sequence: Computes Fibonacci numbers efficiently.
* Longest Common Subsequence: Finds the longest subsequence common to two sequences.
* Knapsack Problem: Optimizes the total value in a knapsack without exceeding its capacity.

**5. Greedy Algorithms**

* Definition: Makes the locally optimal choice at each stage.
* Examples:
* Huffman Coding: Compresses data efficiently.
* Kruskal’s Algorithm: Finds the minimum spanning tree.
* Prim’s Algorithm: Finds the minimum spanning tree.
* Sort: Divides the array into halves, sorts them, and merges them.
  + **Quick Sort**: Divides the array using a pivot, sorts the partitions.

**7. Backtracking**

* Definition: Solves problems incrementally by trying partial solutions and then abandoning them if they are not suitable.
* Examples:
* N-Queens Problem: Places N queens on an N×N chessboard.
* Sudoku Solver: Solves Sudoku puzzles.

**REASON FOR CHOOSING DATA STRUCTURE AND ALGORITHM**

**Importance of DSA in Computer Science**

1. Foundation of Programming: DSA provides the fundamental building blocks for programming. Understanding data structures like arrays, linked lists, stacks, queues, trees, and graphs, along with algorithms for sorting, searching, and optimization, is crucial for writing efficient code.
2. Problem-Solving Skills: DSA enhances problem-solving abilities. It teaches how to break down complex problems into manageable parts and solve them systematically. This skill is invaluable in both academic and professional settings.
3. Performance Optimization: Efficient data structures and algorithms are key to optimizing the performance of software applications. They help in reducing time and space complexity, making programs run faster and use fewer resources.
4. Technical Interviews: Knowledge of DSA is often a prerequisite for technical interviews at top tech companies. Mastery of these concepts can significantly improve the chances of securing a job in the software industry.

**Benefits of a Summer Training Course in DSA**

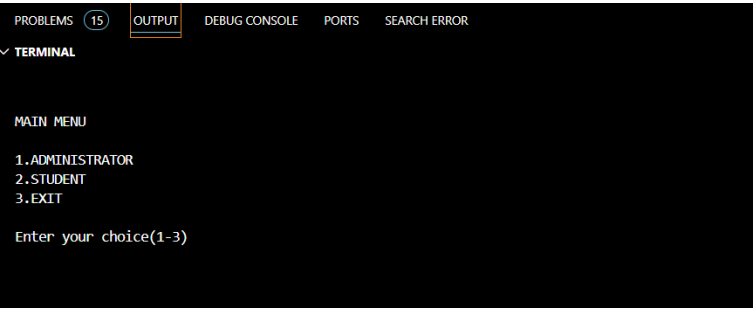
1. Intensive Learning Experience: A summer training course offers an intensive learning experience, allowing students to focus solely on DSA without the distractions of regular coursework. This concentrated effort can lead to a deeper understanding of the subject.
2. Hands-On Practice: Such courses typically include hands-on practice through coding exercises, projects, and real-world problem-solving scenarios. This practical experience is crucial for reinforcing theoretical knowledge.
3. Expert Guidance: Summer training programs often feature instructors who are experts in the field. Their guidance can help students navigate complex topics and provide insights that are not readily available in textbooks.
4. Networking Opportunities: These courses provide opportunities to network with peers and professionals who share similar interests. Building a network can be beneficial for future collaborations and career opportunities.

**Skills Acquired Through DSA Training**

1. Algorithmic Thinking: Students learn to think algorithmically, developing the ability to design and analyze algorithms for various problems. This skill is essential for software development and research.
2. Data Management: Understanding different data structures and their applications helps in managing data efficiently. This is particularly important in fields like data science, machine learning, and big data.
3. Coding Proficiency: Regular coding practice improves proficiency in programming languages such as Python, Java, C++, and others. This is a valuable asset for any software developer.
4. Analytical Skills: DSA training enhances analytical skills, enabling students to evaluate the efficiency of different approaches and choose the best one for a given problem.

**Real-World Applications of DSA**

1. Software Development: DSA is fundamental to software development. Efficient algorithms and data structures are used in everything from simple applications to complex systems like operating systems, databases, and web servers.
2. Data Science and Machine Learning: In data science, DSA is used for data preprocessing, analysis, and visualization. Machine learning algorithms often rely on efficient data structures for training and prediction.
3. Cybersecurity: Algorithms play a crucial role in encryption, decryption, and other security protocols. Understanding DSA is essential for developing secure systems.
4. Game Development: Game developers use data structures and algorithms to manage game states, handle user inputs, and create realistic simulations.



**PROJECT OVERVIEW AND ANALYSIS**

**1. Main Menu Display**

* **Title**: The interface begins with a clear title, "MAIN MENU," which indicates to the user that they are at the primary navigation point of the system.
* **Options**:
  + **1. ADMINISTRATOR**: This option allows users with administrative privileges to access functionalities related to managing the library, such as adding or modifying student and book records.
  + **2. STUDENT**: This option is intended for student users, enabling them to perform actions like viewing available books, issuing books, or returning them.
  + **3. EXIT**: This option provides a way for users to exit the program safely, ensuring that they can terminate their session when desired.

**2. User Prompt**

* **Input Instruction**: The prompt "Enter your choice (1-3)" instructs users to input a number corresponding to their desired action. This clear instruction is crucial for guiding users on how to interact with the menu effectively.

**3. Input Handling**

* **Choice Input**: The program captures the user's input through cin >> ch;, where ch is the variable that stores the user's choice. This input is then processed using a switch statement to determine the next action based on the user's selection.

**4. Error Handling**

* **Invalid Input Response**: If the user enters a number outside the specified range (1-3), the program can respond with an alert (e.g., using cout << "\a";), which produces a beep sound to indicate an invalid choice. This feedback is essential for improving user experience by prompting them to enter a valid option.

**5. Function Calls**

* **Menu Navigation**: Based on the user's choice, the program will call specific functions:
  + If the user selects option 1, the adminmenu() function is invoked, leading to the administrator-specific functionalities.
  + If option 2 is chosen, the studentmenu() function is called, directing the user to student-related operations.
  + Option 3 triggers the program to exit using exit(0);, ensuring a clean termination of the application.

1. **ADMIN MENU:**

* **Title**: The interface begins with the title "ADMIN MENU," indicating that the following options are specific to administrative functions within the library management system.

1. **Menu Options**

The menu presents a list of actions that the administrator can perform, each associated with a number:

1. **CREATE STUDENT RECORD**: Allows the admin to add a new student record to the system.
2. **DISPLAY ALL STUDENTS RECORD**: Displays a list of all student records currently in the system.
3. **DISPLAY SPECIFIC STUDENT RECORD**: Enables the admin to view details of a specific student by entering their identifier (e.g., admission number).
4. **MODIFY STUDENT RECORD**: Allows the admin to update the details of an existing student record.
5. **DELETE STUDENT RECORD**: Provides the option to remove a student record from the system.

A screenshot of a computer

Description automatically generated

1. **CREATE BOOK**: Allows the admin to add a new book to the library's inventory.
2. **DISPLAY ALL BOOKS**: Displays a list of all books available in the library.
3. **DISPLAY SPECIFIC BOOK**: Enables the admin to view details of a specific book by entering its identifier (e.g., ISBN).
4. **MODIFY BOOK**: Allows the admin to update the details of an existing book record.
5. **DELETE BOOK**: Provides the option to remove a book from the library's inventory.
6. **BACK TO MAIN MENU**: This option allows the admin to return to the main menu of the library management system.

**4. User Prompt**

* **Input Instruction**: The prompt "Enter your choice (1-11)" instructs the user to input a number corresponding to their desired action. This clear instruction is crucial for guiding the user on how to interact with the menu effectively.

**1. Input Prompts**

* **Enter The Admission No. 01**: This prompt asks the user to input the admission number for the new student. The format suggests that the user should enter a unique identifier for the student.
* **Enter The Student Name Aarya**: This prompt requests the user to input the name of the student. In this case, "Aarya" is provided as an example input.

**2. Confirmation Message**

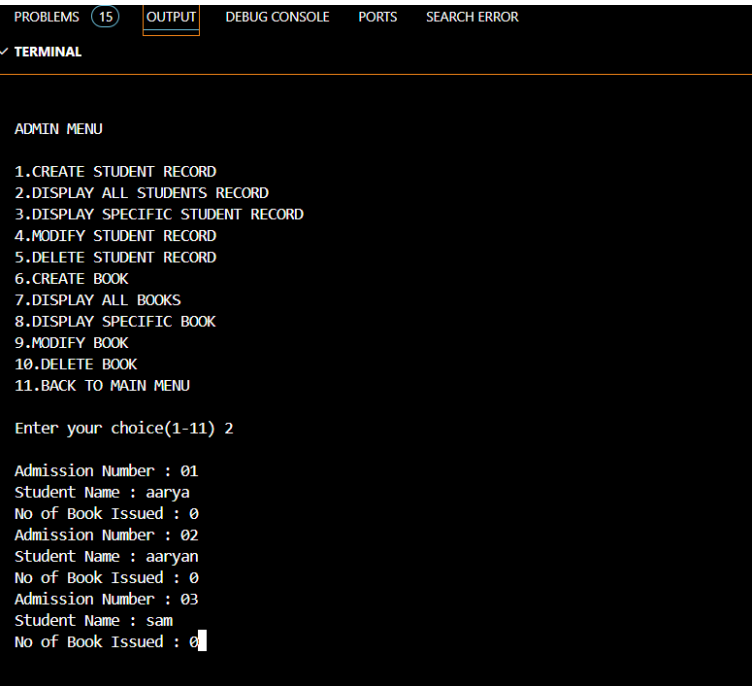
* **Student Record Created...**: This message confirms that the student record has been successfully created in the system. It provides feedback to the user, indicating that their input has been processed correctly.

**3. Follow-Up Prompt**

* **Do you want to add more record...(y/n)?**: This prompt asks the user if they wish to add another student record. The user can respond with 'y' (yes) to continue adding records or 'n' (no) to stop the process.

A screenshot of a computer error

Description automatically generated



**1. Terminal Section**

* **Title**: The interface begins with the title "NEW BOOK ENTRY...", indicating that the user is in the process of adding a new book record.

**2. Input Prompts**

* **Enter The Book ID:** This prompt asks the user to input the unique identifier for the new book. The format suggests that the user should enter a unique code or number for the book.
* **Enter The Book Title:** This prompt requests the user to input the title of the book.
* **Enter The Author's Name:** This prompt asks for the name of the author of the book.
* **Enter The Publisher's Name:** This prompt requests the user to input the name of the publisher.
* **Enter The Price:** This prompt asks for the price of the book.
* **Enter The Quantity:** This prompt requests the user to input the number of copies available for the book.

**3. Confirmation Message**

* **Book Record Created...**: This message confirms that the book record has been successfully created in the system. It provides feedback to the user, indicating that their input has been processed correctly.

**4. Follow-Up Prompt**

* **Do you want to add more records...(y/n)?**: This prompt asks the user if they wish to add another book record. The user can respond with 'y' (yes) to continue adding records or 'n' (no) to stop the process. This feature enhances usability by allowing multiple entries in a single session.

A screenshot of a computer program

Description automatically generated

**1. Terminal Section**

* **Title**: The interface begins with the title "NEW STUDENT ENTRY...", indicating that the user is in the process of adding a new student record.

**2. Input Prompts**

* **Enter The Admission No. 01**: This prompt asks the user to input the admission number for the new student. The format suggests that the user should enter a unique identifier for the student.
* **Enter The Student Name Aarya**: This prompt requests the user to input the name of the student. In this case, "Aarya" is provided as an example input.

**3. Confirmation Message**

* **Student Record Created...**: This message confirms that the student record has been successfully created in the system. It provides feedback to the user, indicating that their input has been processed correctly.

**4. Follow-Up Prompt**

* **Do you want to add more record...(y/n)?**: This prompt asks the user if they wish to add another student record. The user can respond with 'y' (yes) to continue adding records or 'n' (no) to stop the process. This feature enhances usability by allowing multiple entries in a single session.



1. **1. Title: “MODIFY STUDENT RECORD”**
   * This indicates that the application is being used to modify existing student records.
2. **Fields:**

* Admission Number: The admission number of the student being modified is “03”.
* Student Name: The current name of the student.
* No of Book Issued: Indicates the number of books issued to the student, which is “0” in this case.
* Modify Student Name: A prompt to enter the new name for the student.

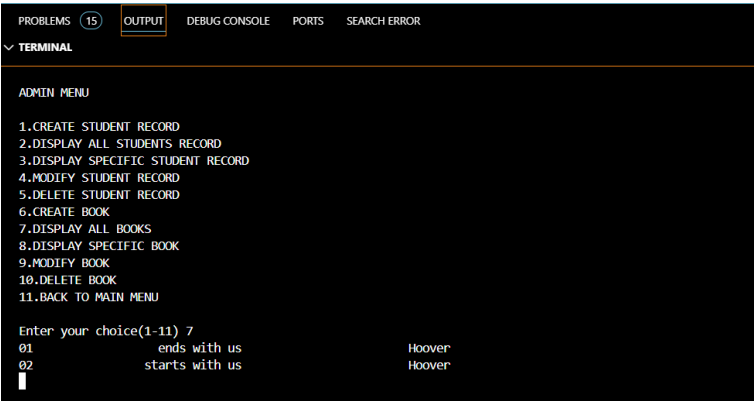
1. **Action:**

* The user has entered the new details for the student with admission number “03”.
  + The message “Record Updated” highlighted in yellow confirms that the student’s record has been successfully updated.

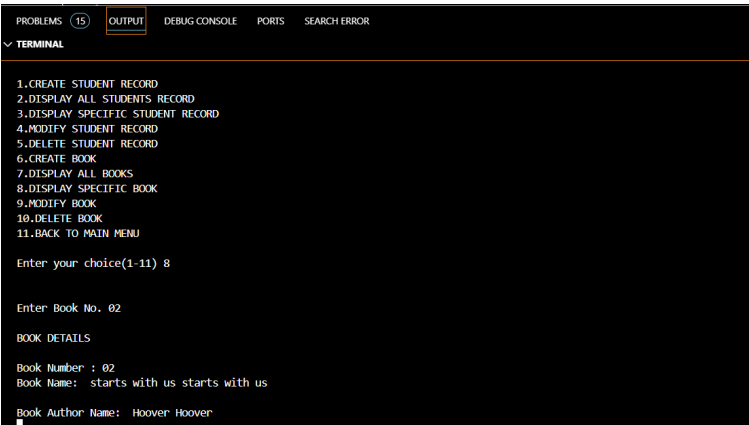
A black screen with white text

Description automatically generated

1. **DELETE STUDENT...**: This indicates that the program is initiating the process to delete a student record.
2. **Enter the Admission no > : 03**: The user is prompted to input the admission number of the student whose record they wish to delete. In this case, the user has entered "03."
3. **Record Deleted..**: This message confirms that the record associated with admission number 03 has been successfully deleted from the system.



1. **Title**: “NEW BOOK ENTRY”
   * This indicates that the user is in the process of adding a new book to the system.
2. **Prompts**:
   * **ENTER BOOK NO.**: The user is prompted to enter the book number, which is “01” in this case.
   * **ENTER BOOK NAME**: The user is prompted to enter the name of the book. The entered book name is “It Ends with Us”.
   * **ENTER AUTHOR NAME**: The user is prompted to enter the author’s name. The entered author name is “Colleen Hoover”.
3. **Action**:
   * After entering the book number, name, and author, the system confirms the creation of the book entry with the message “Book Created …”.
4. **Next Step**:
   * The system then asks if the user wants to add more records with the prompt “Do you want to add more record … (y/n ).” The user has responded with “y” (yes), indicating they want to add another book entry.



**Menu Options**

The interface displays a menu with various options for managing records:

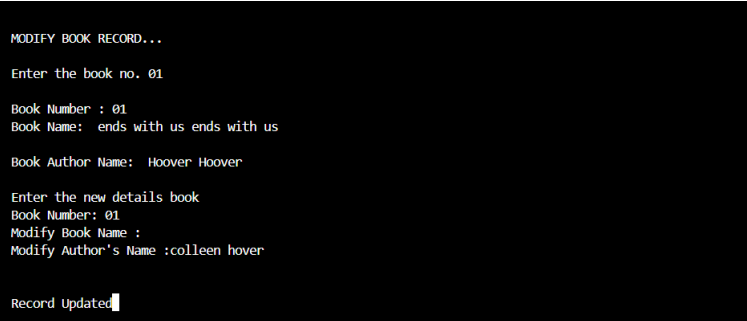
1. **CREATE STUDENT RECORD**: Allows the user to create a new student record.
2. **DISPLAY ALL STUDENTS RECORD**: Displays all student records in the database.
3. **DISPLAY SPECIFIC STUDENT RECORD**: Displays a specific student record based on the admission number.
4. **MODIFY STUDENT RECORD**: Allows the user to modify an existing student record.
5. **DELETE STUDENT RECORD**: Deletes a student record from the database.
6. **CREATE BOOK**: Allows the user to create a new book record.
7. **DISPLAY ALL BOOKS**: Displays all book records in the database.
8. **DISPLAY SPECIFIC BOOK**: Displays a specific book record based on the book number.
9. **MODIFY BOOK**: Allows the user to modify an existing book record.
10. **DELETE BOOK**: Deletes a book record from the database.
11. **BACK TO MAIN MENU**: Returns to the main menu.

**User Interaction**

* The user has chosen option **8** to display a specific book record.
* The system prompts the user to enter the **Book No.**, which is “02” in this case.

**Book Details**

* The system displays the details of the book with **Book Number: 02**.
* **Book Name**: “starts with us starts with us”
* **Book Author Name**: “Hoover Hoover”



1. **Title**: “MODIFY BOOK RECORD”
   * Indicates that the user is in the process of modifying a book record.
2. **User Input**:
   * The user entered the book number as “01”.
3. **Current Book Details**:
   * **Book Number**: 01
   * **Book Name**: “ends with us ends with us”
   * **Author Name**: “Hoover Hoover”
4. **Modification Prompt**:
   * The system asks the user to enter new details for the book.
   * Specifically, it prompts for:
     + **Modify Book Name**: The user entered “colleen hover” as the new book name.
     + **Modify Author’s Name**: Again, the user entered “colleen hover” as the new author’s name.
5. **Confirmation**:
   * The system displays “Record Updated,” indicating that the changes have been saved.

A screenshot of a computer

Description automatically generated

**Menu Options**

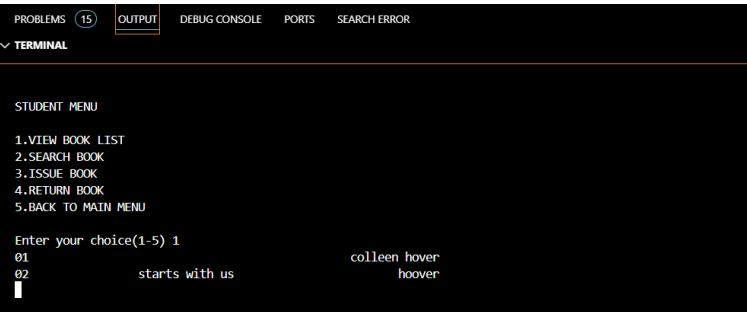
The interface displays a menu with various options for managing records:

1. **VIEW BOOK LIST**: Allows the user to view the list of all books available in the system.
2. **SEARCH BOOK**: Enables the user to search for a specific book in the database.
3. **ISSUE BOOK**: Allows the user to issue a book to a student.
4. **RETURN BOOK**: Enables the user to return a previously issued book.
5. **BACK TO MAIN MENU**: Returns to the main menu of the system.

**User Interaction**

* The user has chosen option **1** to view the book list.
* The system displays the list of books with their details:
  + **Book Number: 01**
    - **Book Name**: “starts with us”
    - **Author Name**: “colleen hover”
  + **Book Number: 02**
    - **Book Name**: “colleen hoover”
    - **Author Name**: “hoover”

1. **Title**: “DELETE BOOK”
   * Indicates that the user is in the process of deleting a book record.
2. **User Input**:
   * The user is prompted to enter the book number to delete. The entered book number is “02”.
3. **Action**:
   * After entering the book number, the system confirms the deletion of the book record with the message “Record Deleted.”



A screenshot of a computer program

Description automatically generated

**Menu Options**

The interface displays a menu with various options for managing book records:

1. **VIEW BOOK LIST**: Allows the user to view the list of all books available in the system.
2. **SEARCH BOOK**: Enables the user to search for a specific book in the database.
3. **ISSUE BOOK**: Allows the user to issue a book to a student.
4. **RETURN BOOK**: Enables the user to return a previously issued book.
5. **BACK TO MAIN MENU**: Returns to the main menu of the system.

**User Interaction**

* The user has chosen option **2** to search for a book.
* The system prompts the user to enter the **Book No.**, which is “02” in this case.

**Book Details**

* The system displays the details of the book with **Book Number: 02**.
* **Book Name**: “starts with us starts with us”
* **Book Author Name**: “hoover hoover”

A screenshot of a computer program

Description automatically generated

**Title**

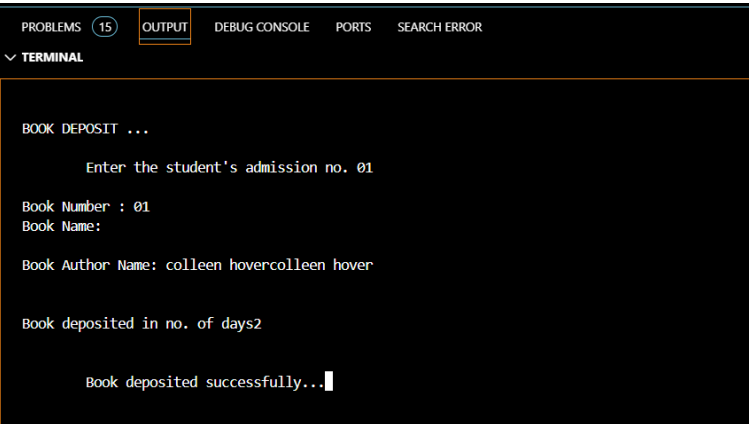
* **BOOK ISSUE**: Indicates that the user is in the process of issuing a book to a student.

**User Input**

1. **Enter the student’s admission no.**: The user has entered “01” as the student’s admission number.
2. **Enter the book no.**: The user has entered “01” as the book number.

**Book Details**

* **Book Number**: 01
* **Book Author Name**: “colleen hovercolleen hover” **Confirmation**
* The system confirms the successful issuance of the book with the message “Book issued successfully … …”



1. **Title**: “BOOK DEPOSIT”
   * Indicates that the user is in the process of depositing a book.
2. **User Input**:
   * The user entered the student’s admission number as “01”.
   * The system then prompts for the book number, which is also entered as “01”.
   * Lastly, the system asks for the number of days the book was deposited, and the user entered “2”.
3. **Book Details**:
   * **Book Number**: 01
   * **Book Name**: It starts with us.
   * **Book Author Name**: “colleen hovercolleen hover” (It seems there is a repetition here, possibly a typo or input error)
4. **Confirmation**:
   * The system confirms the successful deposit of the book with the message “Book deposited successfully … .”

**CONCLUSION**

In conclusion, on analyzing Airbnb data using R programming effectively highlights the intricacies of the short-term rental market. By employing essential packages such as dplyr, tidyr, ggplot2, and lubridate, conducted a thorough exploration of the dataset, addressing critical issues like missing values and data type conversions. The project involved replacing NA values in the reviews\_per\_month column with 0, converting the last\_review column to a date format, and removing rows with incomplete data, ensuring the integrity of the analysis. Through various data transformations, including converting room\_type to a factor and calculating reviews\_per\_year. The visualizations created, including histograms and boxplots, effectively illustrated the distribution of prices and the relationship between room types and pricing. Overall, the project not only demonstrates the proficiency in data analysis but also offers valuable insights for stakeholders in the Airbnb market, emphasizing the importance of data-driven decision-making in understanding market dynamics.

**BIBLOGRAPHY**

1. Board Infinity website: <https://www.boardinfinity.com/learner/my-learning>
2. Udemy: <https://www.udemy.com/course-dashboard-redirect/?course_id=2121018>