Carefully read the following instructions before attempting the paper.

- The Exam paper consists of 3 questions on 2 printed sides of 1 page.
- In case of any ambiguity, you may make assumptions, but your assumption must not contradict any statement in the question paper. Also mention your assumptions.
- **DON'T** share your program, if your code is matched to any member of your class, both will get **straight F** in the course without asking who shared or who magically copied.
- All your files must be named by your roll number along with question number e.g. **K23-XXXX_Q1.cpp.**

Question # 01 (LLO #: 1) ***[20 Points, 15 Weightage]***

A warehouse is managing a shipment process where packages are being processed based on their arrival times. The warehouse uses a dynamic system to track and process incoming packages. Initially, there are a set of package IDs with timestamps indicating their arrival: "pkg1" at 45 minutes, "pkg2" at 20 minutes, "pkg3" at 35 minutes, "pkg4" at 10 minutes, "pkg5" at 50 minutes, "pkg6" at 30 minutes, and "pkg7" at 25 minutes. The warehouse uses a min-heap to manage the packages, ensuring that the package that arrived the earliest is always processed first. As new packages arrive, the heap should be updated to maintain the correct order. After inserting all the initial packages, you need to extract the first three packages to be processed (those that arrived the earliest), ensuring the heap is restructured after each extraction. Once the first three packages are processed, transfer the remaining packages to a stack, and then use the stack to reconfigure the heap to process the packages that arrived the latest.

Example Output:

First 3 Packages to Process (Earliest Arrival): pkg4 pkg2 pkg7

Remaining Packages (Latest Arrival): pkg5 pkg1 pkg3 pkg6

National University of Computer and Emerging Sciences

Question # 02 (LLO #: 2) ***[40 Points, 20 Weightage]***

GreenThumb, a leading nursery and gardening supplies store, has been managing an expanding inventory of plants and gardening products. As the number of products grows, GreenThumb faces challenges in efficiently searching, updating, and managing its product records, especially when dealing with varying prices and plant types. To address this, the store plans to implement a program using an AVL Tree, which will allow for efficient insertion, deletion, and searching of plant records based on their prices. The system will store product details, including plant names, categories (e.g., indoor, outdoor, succulents), and their retail prices. By organizing the data in an AVL Tree, GreenThumb can quickly search for products within a specified price range, helping customers find plants that meet their budget and gardening needs. The program will also allow the store to delete records of products based on their prices (e.g., removing outdated stock or discounted items), helping manage inventory more effectively. This system will enable GreenThumb to streamline its operations, optimize stock levels, and make better purchasing and sales decisions, ultimately providing a more efficient and customer-focused shopping experience.

Question # 03 (LLO #: 3) ***[40 Points, 15 Weightage]***

A university is looking to implement an automated system for managing student exam scores, aiming to streamline the process of storing, retrieving, and sorting exam data. Each student in the system is assigned a unique identifier, **StudentID**, **StudentName**, and an **associated exam score**, which needs to be efficiently stored for future analysis and querying. Given the increasing volume of student data, the university seeks a solution that can handle large datasets, ensure fast access to individual records, and provide effective sorting of student scores. The system should utilize a hash table to store the **StudentID** and corresponding **exam score** pairs. The system will need to automatically resize and rehash the table whenever new students are added. This dynamic resizing will ensure that the system maintains fast access and insertion times even as the number of students and their exam scores increases.

Once the student data is stored in the hash table, the system should then be capable of efficiently sorting the students by their exam scores. The sorting should be done in descending order, with higher exam scores appearing first. However, if two students have the same score, the system must then sort them by their **StudentID** in ascending order to maintain a consistent and predictable order for students with identical exam results. This system will enable the university to handle an increasing number of students, track their performance over time, and easily retrieve or update individual records while ensuring data is sorted and accessible in a logical manner for reporting, analysis, or further processing.

Note: In case of collisions, resolve them using the separate chaining method only. For sorting, you can only use QUICK SORT.