Portfolio – Smart Parcel Sorting System Hams Aljohani

System Overview

The Smart Parcel Sorting System is designed to streamline and automate the organization of parcels based on delivery routes and priority levels. It simulates a logistics hub where parcels are received, sorted, and prepared for dispatch using appropriate data structures for efficient processing.  
  
My responsibility in this project focused on implementing a stack for reversing batches of parcels. This functionality is essential when a group of parcels needs to be dispatched in the reverse order they were received, often due to last-minute changes in routing or delivery scheduling.

Problem Description

In real-world parcel distribution systems, certain circumstances require reversing the delivery order of a batch of parcels. For instance, if the route of a delivery vehicle changes or urgent packages are added, the order of processing needs to be inverted. My task was to design and implement a solution to reverse a batch of parcels efficiently using a stack, which naturally supports LIFO (Last-In-First-Out) behavior.

Architecture

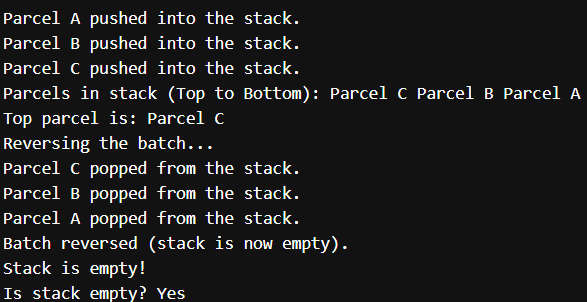
The system architecture includes multiple components working together to process parcels:  
- A queue structure is used for incoming parcels to maintain the original arrival sequence.  
- Arrays and linked lists are used for storing and organizing sorted parcels.  
- Stacks are used to reverse batches of parcels when needed.

Data Structure Usage

the stack was implemented using a linked list to allow flexible, dynamic memory allocation. This approach makes the stack suitable for variable-sized parcel batches. Each node in the stack holds a parcel name and a pointer to the next parcel. This structure supports core stack operations including push, pop, peek, and isEmpty, along with a display function to visualize the current stack and a custom function to reverse an entire batch by popping all items.

My Contribution: Stack for Reversing Batches

I implemented the 'ParcelStack' class using a linked list-based approach. The class includes:  
- `push()`: Adds a new parcel to the top of the stack.  
- `pop()`: Removes the top parcel from the stack.  
- `peek()`: Returns the parcel at the top without removing it.  
- `isEmpty()`: Checks whether the stack is empty.  
- `display()`: Prints all parcels in the stack from top to bottom.  
- `reverseBatch()`: Pops and displays all parcels to simulate batch reversal.

Code snapshot

GitHub Repository

<https://github.com/Hams770/data-project-hams>