**User Guide: Zip Code Group Project 2.0**

**Course Information**

**Course:** CSCI 331 – Software Systems — Fall 2025  
**Project:** Zip Code Group Project 2.0  
**Authors:** Minh Quan Tran, Abel S. Asfaw, Carson Kariniemi, Mitchell Rogers, Mahad Farah  
**Last Updated:** October 2025

**1. Overview**

This program processes U.S. postal code data to demonstrate structured file I/O, buffer manipulation, indexing, and record-based access in C++.  
It performs two primary operations:

1. **Part I – Sequential Processing**
   * Reads and processes postal code data from .csv and **length-indicated** files.
   * Displays formatted postal code tables sorted by ZIP code or by state.
2. **Part II – Indexed Search**
   * Builds and loads a **primary key index** file for efficient lookup.
   * Allows users to retrieve specific ZIP code records directly from the command line.

The program is structured around modular classes:  
HeaderRecordPostalCodeItem, HeaderRecordPostalList, readHeaderRecordPostalCodeBuffer, and indexing components defined in make\_index.cpp.

**2. File Requirements**

Before running the program, ensure the following files are present in the same directory as the executables:

|  |
| --- |
| us\_postal\_codes\_length\_indicated\_header\_record.txt |
| us\_postal\_codes\_ROWS\_RANDOMIZED\_length\_indicated\_header\_record.txt |

**3. Running the Program**

**⚙️ Compilation**

Use the standard C++ compiler to build the executables.

*g++ main3.cpp -o project\_part1.exe*

or

*g++ main4.cpp -o project\_part1.exe*

and

*g++ make\_index.cpp -o project\_part2.exe*

This will create two executables:

* project\_part1 — Sequential processing (row randomized header record postal) (Part I)
* project\_part2 — Indexed search (Part II)

**4. Usage Instructions**

**Part I – Sequential Processing**

**Running with the row randomized header record postal code**

./project\_part1.exe

**Description:**

* Loads the file us\_postal\_codes\_length\_indicated\_header\_record.txt
* Displays all postal records sorted by ZIP code

**Expected Output Example:**

A table of all the postal sorted by zip:

Zip Code Place Name State County Latitude Longitude

-----------------------------------------------------------------------------------------------

00601 Adjuntas PR Adjuntas County 18.1803 -66.7516

00602 Aguada PR Aguada County 18.3795 -67.1887

...

**Description:**

* Loads the row-randomized file us\_postal\_codes\_ROWS\_RANDOMIZED\_length\_indicated\_header\_record.txt
* Ensures equivalent functional output regardless of record order
* Demonstrates buffer handling consistency

**Part II – Indexed Search**

**a. Building and Using the Index**

./project\_part2.exe -Z56301 -Z55101 -Z99999

**Description:**

* Loads or builds indexfile.bin from the length-indicated data file.
* Searches for each ZIP code specified with -Z flag.
* Displays complete labeled output for found ZIPs and a not-found message otherwise.

**Expected Output Example:**

Loaded existing index file: indexfile.bin (41802 entries)

Zip Code: 56301

Place Name: Saint Cloud

State: MN

County: Stearns

Lat: 45.533

Long: -94.1719

55101 not found.

99999 not found.

If indexfile.bin is missing, the program will automatically generate it:

Index not found, building new one...

Index built and written to indexfile.bin (41802 entries)

**5. Command-Line Options**

| **Flag** | **Usage** | **Description** |
| --- | --- | --- |
| -Z##### | Example: -Z56301 | Searches for a specific ZIP code record |
| Multiple -Z flags | Example: -Z56301 -Z55101 -Z99999 | Performs multiple lookups in one run |
| *(No flags)* | Example: ./project\_part2 | Only loads/creates the index without performing searches |

**6. Program Output Summary**

| **Component** | **Output Type** | **Description** |
| --- | --- | --- |
| Part I (main3.cpp, main4.cpp) | Tabular Output | Prints all postal data in a formatted table |
| Part II (make\_index.cpp) | Search Results | Prints individual ZIP records with labeled fields |
| Errors | Console Message | Displays warnings for missing files or invalid ZIPs |

**7. Expected Directory Structure**

/ProjectDirectory

├── main1.cpp

├── main2.cpp

├── main3.cpp

├── main4.cpp

├── make\_index.cpp

├── PostalList.cpp

├── PostalList.h

├── PostalCodeItem.cpp

├── PostalCodeItem.h

├── HeaderRecordPostalList.cpp

├── HeaderRecordPostalList.h

├── HeaderRecordPostalCodeItem.cpp

├── HeaderRecordPostalCodeItem.h

├── readPostalCodeBuffer.cpp

├── readHeaderRecordPostalCodeBuffer.cpp

├── us\_postal\_codes\_length\_indicated\_header\_record.txt

├── us\_postal\_codes\_ROWS\_RANDOMIZED\_length\_indicated\_header\_record.txt

├── us\_postal\_code.csv

├── us\_postal\_code\_ROW\_RANDOMIZED.csv

└── indexfile.bin

**8. Troubleshooting**

| **Issue** | **Possible Cause** | **Solution** |
| --- | --- | --- |
| “Error: unable to open file” | Missing CSV or text file | Ensure files are in the same directory |
| “not found” message for ZIP | ZIP not present in dataset | Verify ZIP exists in the source CSV |
| Garbled characters in output | Wrong encoding or newline type | Ensure files use UTF-8 and UNIX-style line endings |
| Index doesn’t update | Index file cached | Delete indexfile.bin to rebuild |

**9. Notes**

* The program intentionally restricts memory usage to **one record at a time** (per project requirements).
* Doxygen documentation is generated from the inline comments and headers (.h and .cpp files).
* Ensure that all CSV and TXT files are plain-text formatted (no Excel metadata).

Thank You!!