	Bansilal Ramnath Agarwal Charitable Trust's Vishwakarma Institute of Information Technology **Department of** **Artificial Intelligence and Machine Learning**	
Name: Yuvraj Prabhakar Sankilwar		
Class: TY	Division: A	Roll No: 391052
Semester: Vl	Academic Year:2024-2025	
Subject Name & Code: ECI3120G - Mainframe Technologies		
Title of Assignment: To write a REXX program that removes duplicate records from a 100-byte fixed-length PS file and sorts the unique records based on a primary key in columns 25–32.		
Date of Performance: 05/04/2025	Date of Submission: 05/04/2025	

Aim:

To write a REXX program that removes duplicate records from a 100-byte fixed-length PS file and sorts the unique records based on a primary key in columns 25–32.

Problem Statement:

This assignment focuses on fundamental data processing tasks in a mainframe environment using REXX (Restructured Extended Executor) scripting. The primary objectives are:

- Creation of a **Physical Sequential (PS)** dataset with structured records.
- Elimination of **duplicate records** based on the full content or a subset of the content.
- **Sorting** records based on a defined **primary key**, extracted from specific columns

Background Information:

The dataset used in this assignment is a **PS (Physical Sequential)** dataset, which is one of the simplest forms of datasets used in z/OS systems. Each record in this dataset is:

- **100 bytes** long
- **Fixed Blocked (FB)** format (meaning each line has the same length)
- Contains a **primary key** in **columns 25 to 32** (i.e., 8-character keys)

Out of **120 total records**, the distribution is as follows:

- **105 unique records**
- **15 duplicate records** (exact replicas of earlier records)

- **Randomized order**

This simulates a raw data input scenario where duplicates and unsorted data are common — especially in log files, transactional feeds, or input from third-party systems.

Key Concepts Covered

1. REXX Programming Language

REXX is a scripting language widely used in IBM mainframes. It is valued for its readability, string manipulation capabilities, and integration with z/OS components.

For this assignment, REXX is used due to:

- Simplicity in handling file I/O
- Built-in support for string operations (parsing, trimming, substring)
- Ease of debugging and prototyping

2. File Reading and Writing in REXX

Reading a PS file in REXX involves using the EXECIO command, which allows batch reads and writes from/to datasets:

- EXECIO * DISKR ddname (FINIS – reads the entire dataset into a stem variable
- EXECIO * DISKW ddname (FINIS – writes a stem variable back to a dataset

3. Duplicate Detection and Removal

In the context of this assignment, **duplicate detection** involves:

- Comparing entire lines (or optionally a substring)
- Tracking which lines have already been seen
- Writing only the unique ones to the output

A common technique is to use a **REXX stem variable as a hash table**:

```
if duplicates.line = '' then do
  duplicates.line = 1
  output_lines.count = line
end
```

4. Primary Key Extraction and Sorting

The **primary key** is located between **column 25 and 32**. REXX uses the substr(string, start, length) function to extract substrings:

```
key = substr(record, 25, 8)
```

To sort records by this key, a multi-step approach is used:

- Extract the key
- Store each record along with its key in a stem array
- Use REXX’s SORT or manually implement a sorting algorithm like **bubble sort** or **simple selection**

- **sort** (for small datasets like 120 records)

Alternatively, in z/OS environments, sorted records could also be handled using **DFSORT** or **SYNCSORT**, but for the purpose of this assignment, the logic is implemented purely in REXX.

Program Flow: Step-by-Step

Step 1: Read the Dataset

Use EXECIO to read all 120 records into a REXX stem:

```
"ALLOCATE F(DATAIN) DA('Z66845.REXX.INPUT') SHR REUSE"
"EXECIO * DISKR DATAIN (STEM LINES. FINIS)"
```

Step 2: Remove Duplicates

Maintain a secondary stem (e.g., UNIQUE.) and hash to track already seen records:

```
do i = 1 to LINES.0
  line = LINES.i
  if !seen.line then do
    seen.line = 1
    UNIQUE.count = line
  end
end
```

Step 3: Sort by Primary Key

For each record:

- Extract the key (columns 25–32)
- Store it in an indexed format (e.g., KEY.i = substr(UNIQUE.i, 25, 8))
- Sort the records by comparing the keys (simple sort)

Sorting example (bubble sort style):

```
do i = 1 to N-1
  do j = i+1 to N
    if key.i > key.j then
      /* swap records */
  end
end
```

Step 4: Write Sorted Records to Output Dataset

```
"ALLOCATE F(DATAOUT) DA('Z66845.REXX.SORTED') SHR REUSE"
"EXECIO * DISKW DATAOUT (STEM SORTED. FINIS)"
```

Expected Output

After running the REXX program:

- The duplicates (15 records) are removed
- 105 records remain
- These records are sorted in **ascending order of the primary key (columns 25–32)**

- Output is saved to a dataset like Z66845.REXX.SORTED

Practical Significance

This exercise demonstrates real-world data manipulation on mainframes:

- Dealing with inconsistent data
- Cleaning and structuring for further processing
- Manual implementation of basic algorithms in a low-level scripting language

Such skills are vital for jobs in **system programming**, **data engineering**, **ETL development**, and **mainframe support roles**.

It also reinforces:

- Efficient handling of **sequential datasets**
- Resourceful use of memory (using REXX's stem variables instead of arrays or files)
- Understanding of data formats (LRECL, RECFM, BLKSIZE)

Github Repo Link:

https://github.com/yuvrajofficials/mainframe-assignments.git

Conclusion:

This assignment blends concepts of data structure, mainframe file handling, and REXX scripting. Through this task, students strengthen their ability to manipulate fixed-format datasets, eliminate redundant entries, and apply algorithmic thinking for sorting. These concepts extend to broader areas such as database deduplication, sorted report generation, and batch data preparation — making this not only an academic exercise but also a valuable professional skill.