### Question 1)

### a)

### Ascii file 320MB

### Binary file 122MB

b) Estimation of Array Size in Memory

* **Matrix Dimensions**: n=4000 so the matrix has n\*n = 4000\*4000 = 16,000,000 elements
* **Size of one double**: 8  bytes
* **Size in memory**: Memory size = 8 × n\*n = 8 × 16,000,000 = 128 MB.
* **Memory size** = 8 × n\*n = 8 × 16,000,000 = 128MB.

Size in disk for ASCII format : 320 MB

Size in disk for Binary format : 128 MB

As we saw **Binary** format is taking less space than **ASCII** so it is better to store data in **Binary** format as it is space efficient.

**Binary Format**:

Smaller file size (33% smaller compared to CSV for n = 4000).

Faster read/write performance .

Scales better for very large datasets.

**Code Explanation :**

This program generates a n×n matrix, populates it with values, and writes it to disk in two formats: **ASCII (text)** and **binary**.

#### main() ****Function****

**Matrix Allocation**:

Dynamically allocates memory for a 2D matrix A of size n×n.

Ensures memory is allocated row by row using malloc. If allocation fails, an error message is printed.

**Matrix Initialization**:

Populates the matrix A with values: A[ i ][ j ] = i + j

**File Writing**:

Calls the print\_to\_file function twice:

format\_flag = 0: Writes the matrix in **ASCII (text)** format.

format\_flag = 1: Writes the matrix in **binary** format.

**Memory Deallocation**:

Frees the memory allocated for the matrix to prevent memory leaks.

#### print\_to\_file() ****Function****

Creates a filename using the matrix size (n) and format:

array\_<size>\_asc.out for ASCII format.

array\_<size>\_bin.out for binary format.

**File Opening**:

Opens the file in either **write mode (**w**)** for ASCII or **binary write mode (**w**)** for binary format.

**Matrix Writing**:

**ASCII Format** (format\_flag == 0):Writes matrix elements row by row as floating-point numbers with 15 decimal precision.

**Binary Format** (format\_flag == 1):Writes each row of the matrix directly using fwrite to store raw binary data.

#### ****Key Takeaways****

**Efficiency**:

Binary format is faster and takes less storage space but is not human-readable.

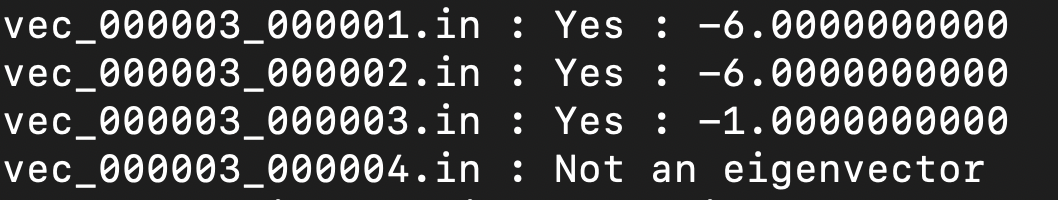
ASCII format is easier to inspect and debug but less efficient for large data.

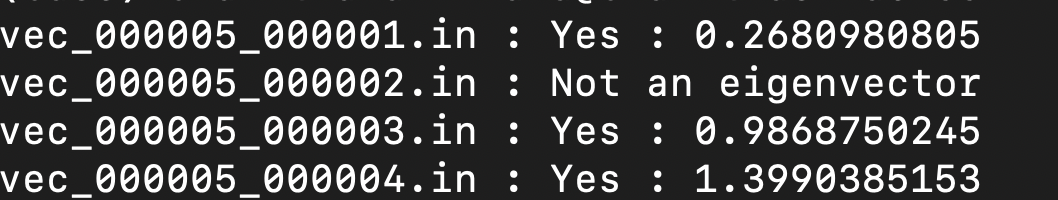
**Scalability**:

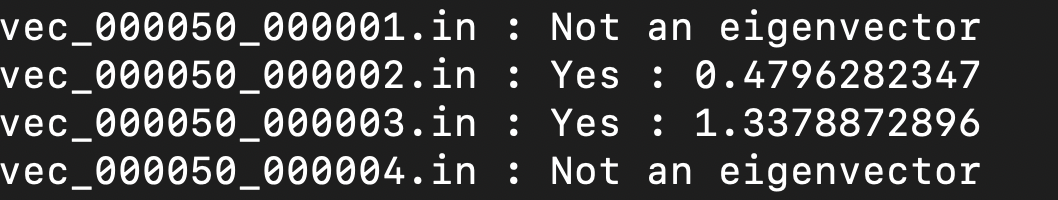
The program scales well for large matrices like n = 4000 , given sufficient memory.

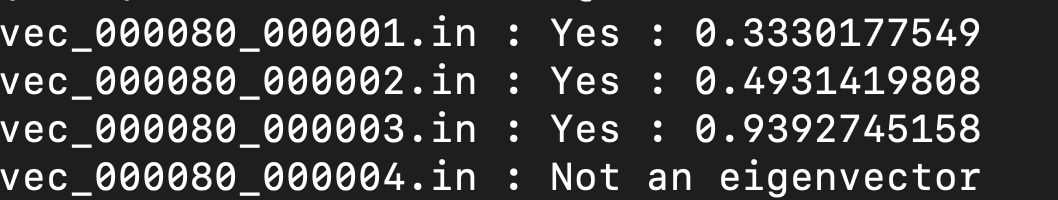
### Question 2)

### These are the outputs of the given inputfiles









**Code Explanation :** Finds whether the given vector is a eigen vector for the given matrix.If so it’s corresponding eigen value is printed in the screen or else it prints that “Not an eigen -vector”

### main

* Reads the matrix size (n) from an input file.
* Dynamically allocates memory for the matrix and vector.
* Loops through all available vector files for the given matrix and processes each one using is\_eigenvector function.

### read\_matrix

* Reads a matrix from a CSV file.
* Validates dimensions and ensures all data is correctly formatted.

### read\_vector

* Reads a vector from a file, handling whitespace and potential formatting errors.

### ****Purpose****

The is\_eigenvector function determines whether a given vector v is an eigenvector of a matrix A and calculates the corresponding eigenvalue λ if it is.

### ****Process****

**Compute** Av:

Multiplies the matrix A with vector v to produce a new vector A\*v.

**Scaling for Numerical Stability**:

Identifies the maximum absolute component of A\*v to avoid overflow or underflow.

Scales both A\*v and v by this maximum if it exceeds a large threshold.

**Identify the Eigenvalue**:

Compares corresponding elements of Av and v where v[i] ≠ 0

Computes λ=Av[i] /v[i]​ using the most numerically stable component.

**Verification**:

Checks if Av≈λv within a small tolerance (EPSILON).

Considers numerical precision errors and ensures consistency across all components.

**Return Result**:

Returns 1 (true) if v is an eigenvector and stores the eigenvalue.

Returns 0 (false) if the conditions are not satisfied.