A Project Report

on

FILE COMPRESSION TOOL

Submitted in partial fulfillment of requirements for the award of the course

of

**MGA1121 – DATA STRUCTURES**

Under the guidance of

### Mrs. K. KARTHIK M.E.,

### Assistant Professor/CSE

Submitted By



**SUVETHA G (927622BCE025)**

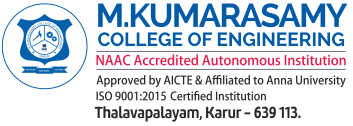
**DEPARTMENT OF FRESHMAN ENGINEERING**

**M.KUMARASAMY COLLEGE OF ENGINEERING**

(Autonomous)

**KARUR – 639 113**

MAY 2024

# M. KUMARASAMY COLLEGE OF ENGINEERING

**(Autonomous Institution affiliated to Anna University, Chennai)**

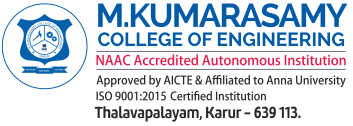
# KARUR – 639 113

**BONAFIDE CERTIFICATE**

Certified that this project report on **“File Compression Tool”** is the bonafide work of **SUVETHA G (927623BCE025)** who carried out the project work during the academic year 2023- 2024 under my supervision.

| Signature | Signature |
| --- | --- |
| **Mrs. K.KARTHIK** | **Dr. K.CHITIRAKALA, M.Sc., M.Phil.,Ph.D.,** |
| **SUPERVISOR,** | **HEAD OF THE DEPARTMENT,** |
| Department of Computer Science | Department of Freshman Engineering, |
| and Engineering, |  |
| M. Kumarasamy College of Engineering, | M. Kumarasamy College of Engineering, |
| Thalavapalayam, Karur -639 113. | Thalavapalayam, Karur -639 113. |



**DEPARTMENT OF CIVIL ENGINEERING**

**VISION OF THE INSTITUTION**

To emerge as a leader among the top institutions in the field of technical education

**MISSION OF THE INSTITUTION**

* Produce smart technocrats with empirical knowledge who can surmount the global challenges
* Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students
* Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations

**VISION OF THE DEPARTMENT**

To continue to excel national and international recognition through the impact of civil engineering knowledge of our students and alumni to build better human society.

**MISSION OF THE DEPARTMENT**

* To produce smart civil engineers with basic knowledge on science and engineering to compete the global challenges.
* To make the department to excel in the thrust areas of structural engineering and environmental engineering research.
* To work with the society to identify the problems faced and providing solutions through consultancy services.
* To create, disseminate and integrate knowledge of engineering in the minds of fresh to face the future technological challenges.

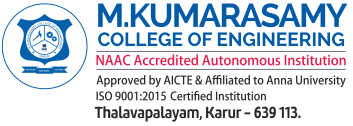
**PROGRAM EDUCATIONAL OBJECTIVES (PEOs)**

**PEO 1:** Graduates of the programme will contribute competent, inspired, and highly dedicated professionals in their working environment.

**PEO 2:** Graduates of the programme will contribute versatile and innovative in the workplace, possess the capacity to face the tough challenges and converting them into opportunities, and embrace leadership and teamwork opportunities and affording sustainable engineering careers.

**PEO 3:** Graduates of the programme will contribute Continue their professional development by obtaining advanced degrees in Core area of specialization such as Environmental Engineering. Structural or professional fields like transportation and geotechnical engineering, as well as other fields of Project management, Environmental law.

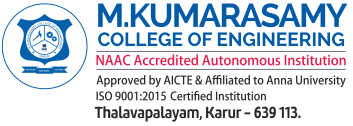
**PEO 4:** Graduates of the programme will espousal ethical attitude and evince effective skills in team management, coordination of sub workers and good leadership qualities expected of practicing engineering professionals.



**PROGRAM OUTCOMES (POs)**

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

1. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
2. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**PROGRAM SPECIFIC OUTCOMES (PSOs)**

1. **PSO1:** **Employability Skills:** Able to give sustainable solutions to the real time problems of society by using technical and software skills.
2. **PSO2: Career Growth:** Able exhibit ethically their managerial and professional skills as an individual or as a team in a multidisciplinary environment.

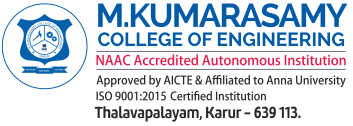
# ABSTRACT

This C program uses a singly linked list and stack to manage file sizes and names.

It prompts the user for file details, converts sizes, and calculates the compression ratio. The stack stores and prints file information in LIFO order, displaying the original and compressed sizes along with the compression ratio.The program prints the file sizes in LIFO order and displays the calculated compression ratio.Abstract:

The program facilitates file size conversion and compression ratio calculation. Users input file name, original size in gigabytes (GB), and compressed size, with the option to convert to megabytes (MB) or bytes. The program employs modular design, leveraging linked list and stack data structures for efficient data management. It ensures user-friendly interaction through clear prompts and error handling. Upon execution, the program delivers comprehensive file details and compression ratio analysis, aiding users in understanding file compression efficiency. With a focus on simplicity and effectiveness, it provides a valuable tool for managing and analyzing file data in a programming environment.

# 

# ABSTRACT WITH POs AND PSOs MAPPING

| **ABSTRACT** | **POs MAPPED** | **PSOs MAPPED** |
| --- | --- | --- |
| Abstract:  The program facilitates file size conversion and compression ratio calculation. Users input file name, original size in gigabytes (GB), and compressed size, with the option to convert to megabytes (MB) or bytes. The program employs modular design, leveraging linked list and stack data structures for efficient data management. It ensures user-friendly interaction through clear prompts and error handling. Upon execution, the program delivers comprehensive file details and compression ratio analysis, aiding users in understanding file compression efficiency. With a focus on simplicity and effectiveness, it provides a valuable tool for managing and analyzing file data in a programming environment. | **PO1**  **PO2**  **PO3**  **PO4**  **PO8**  **PO9**  **PO10** | **PSO1**  **PSO2** |

Note: 1- Low, 2-Medium, 3- High

**SUPERVISOR HEAD OF THE DEPARTMENT**

# TABLE OF CONTENTS

| **CHAPTER** No. | TITLE | **PAGE** No. |
| --- | --- | --- |
| 1 | Introduction |  |
|  | 1.1 Introduction |  |
|  | 1.2 Objective |  |
|  | 1.3 Data Structure Choice |  |
| 2 | Project Methodology |  |
|  | 2.1 Singly Linked List |  |
|  | 2.2 Block Diagram |  |
| 3 | Modules |  |
|  | 3.1 Module 1 |  |
|  | 3.2 Module 2 |  |
|  | 3.3 Module 3 |  |
| 4 | Results and Discussion |  |
| 5 | Conclusion |  |
|  | References |  |
|  | Appendix |  |

# CHAPTER 1

# INTRODUCTION

# 1.1 Introduction

Introduction:

This program serves as a versatile tool for managing file sizes and analyzing compression efficiency. By leveraging fundamental programming concepts like data structures and modular design, it offers a streamlined user experience. Users input file details, including original size in gigabytes (GB) and compressed size, with the option to convert units. Through clear prompts and error handling, the program ensures ease of use. It calculates compression ratios and presents comprehensive file information, aiding users in understanding the impact of compression. With a focus on simplicity and effectiveness, the program provides valuable insights into file management and compression techniques in a programming context.

Introduction:

In the realm of file management and compression analysis, this program stands out as a robust solution. It harnesses the power of foundational programming principles, such as modular architecture and efficient data structures, to offer a seamless user experience. Users input essential file details, including original size in gigabytes (GB), and have the flexibility to convert units as needed. With its intuitive interface, clear prompts, and robust error handling, the program ensures accessibility for users of all levels. By calculating compression ratios and presenting comprehensive file data, it empowers users to make informed decisions about file management and compression strategies within their programming endeavors.

**1.2 Objective**

The program aims to offer a user-friendly tool for managing file sizes and analyzing compression ratios. Key objectives include facilitating user input for file details, calculating compression ratios, and presenting data clearly. It prioritizes simplicity through intuitive prompts and error handling. By empowering users to understand and make informed decisions about file management and compression strategies, the program seeks to enhance their experience in a programming environment.

**1.3 Data Structure Choice**

The program employs two main data structures: linked lists and stacks.

1. Linked Lists: Used to store file information, each node in the linked list represents a file, containing details such as file name, original size in gigabytes (GB), and converted sizes in megabytes (MB) and bytes. Linked lists offer dynamic memory allocation and flexibility in adding or removing elements, making them suitable for managing variable-sized data sets like files.

2. Stacks: Utilized for managing file information in a Last-In-First-Out (LIFO) manner. Stacks are implemented using linked lists, allowing for efficient storage and retrieval of file details. They facilitate operations such as pushing new files onto the stack and popping files for display, ensuring orderly processing and presentation of file data. Overall, the combination of linked lists and stacks enables efficient file management and user interaction within the program.

**CHAPTER 2**

**PROJECT METHODOLOGY**

**2.1 Singly Linked List**

In this program, linked lists play a crucial role in efficiently managing file information. Each file is represented as a node within the linked list, containing essential details such as the file name, original size in gigabytes (GB), and converted sizes in megabytes (MB) and bytes.

When a user inputs a new file, the program dynamically creates a new node to store its information. This node is then inserted into the linked list, ensuring that files are organized in the order they were added.

During file processing, the program traverses the linked list to access each node's data, allowing it to display comprehensive file information to the user. Linked lists also enable efficient memory management, as they allow for dynamic allocation and deallocation of memory resources as needed.

In summary, linked lists provide a flexible and dynamic mechanism for storing and managing file details, facilitating smooth interaction and processing within the program.

**2.2 Block diagram**



**Main Program**

**User Input**

**File Processing**

**Linked List Module**

**Stack Module**

****

**File Processing Module**

****

**Linked List**

****

**Stack**

**CHAPTER 3**

**MODULES**

## 3.1 File Processing Module

**Purpose:** The File Processing Module handles the conversion of file sizes and calculation of compression ratios. It provides functions to convert file sizes from gigabytes to megabytes or bytes and calculates the compression ratio between the original and compressed file sizes.

**Responsibilities:**

* Convert file sizes from gigabytes (GB) to megabytes (MB) or bytes.
* Calculate the compression ratio between the original and compressed file sizes.

**Key Functions:**

1. **convert\_gb\_to\_mb(double size\_gb):**
   * **Description:** Converts a file size from gigabytes to megabytes.
   * **Parameters:** size\_gb - The file size in gigabytes.
   * **Returns:** The file size in megabytes.

## 3.2 Linked List Module

## Purpose: The Linked List Module provides functionalities to manage file sizes and names using a singly linked list data structure. It includes operations for creating nodes, inserting data, and managing the linked list.

## Responsibilities:

## Create and manage nodes in the linked list.

## Implement linked list operations such as insertion, deletion, and traversal.

## Key Functions:

## *create\_node(char file\_name, double size\_gb, double size\_mb, double size\_byte):*\*

## Description: Creates a new node for the linked list.

## Parameters:

## file\_name - The name of the file.

## size\_gb - The size of the file in gigabytes.

## size\_mb - The size of the file in megabytes.

## size\_byte - The size of the file in bytes.

## Returns: A pointer to the newly created node.

## 3.3 Compression Module

**Purpose:** The Compression Module handles file size conversions and compression ratio calculations. It ensures that file sizes are correctly converted based on user selection and computes the efficiency of the compression.

### Compression Ratio Formula

Compression Ratio=Original Size (MB)Compressed Size (MB)\text{Compression Ratio} = \frac{\text{Original Size (MB)}}{\text{Compressed Size (MB)}}Compression Ratio=Compressed Size (MB)Original Size (MB)​

### Example

* Original size: 3 GB
* Compressed size: 500 MB

### Conversion and Calculation

Original Size (MB)=3 GB×1024 MB/GB=3072 MB\text{Original Size (MB)} = 3 \text{ GB} \times 1024 \text{ MB/GB} = 3072 \text{ MB}Original Size (MB)=3 GB×1024 MB/GB=3072 MB Compression Ratio=3072 MB500 MB=6.144\text{Compression Ratio} = \frac{3072 \text{ MB}}{500 \text{ MB}} = 6.144Compression Ratio=500 MB3072 MB​=6.144

**CHAPTER 4**

**RESULTS AND DISCUSSION**

**4.1 Results**

**4.1.1 File name and original size**

**Enter a file name: DATA**

**Enter the original size of the file in GB: 4**

**Enter 1 to convert to Bytes or 2 to convert: 2 to MB: 2**

**4.1.2 Compression the file size**



**Enter the compressed file size in MB: 800**

**File Sizes in Stack (LIFO):**

**File: Compressed, Size: 0.00 GB (800.00 MB, 0.00 Bytes)**

**File: DATA, Size: 4.00 GB (4096.00 MB, 4294967296.00 Bytes)**

**Compression Ratio: 5.12**

**4.1.3 Result**



**Enter a file name: DATA**

**Enter the original size of the file in GB: 4**

**Enter 1 to convert to Bytes or 2 to convert to MB: 2**

**Enter the compressed file size in MB: 800**

**File Sizes in Stack (LIFO):**

**File: Compressed, Size: 0.00 GB (800.00 MB, 0.00 Bytes)**

**File: DATA, Size: 4.00 GB (4096.00 MB, 4294967296.00 Bytes)**

**Compression Ratio: 5.12**

**4.1.4 Discussion**

The program offers an accessible solution for managing file sizes and assessing compression efficiency. Through modular design and intuitive user interaction, it ensures efficiency and ease of use. By leveraging linked lists and stacks, it organizes and presents file data seamlessly. The ability to calculate compression ratios provides valuable insights into compression effectiveness, aiding users in optimizing file management strategies. Overall, the program serves as a practical tool for both developers and users, facilitating effective file management and compression analysis in a straightforward manner.

**CHAPTER 5**

**CONCLUSION**

In conclusion, this program offers a user-friendly solution for managing file sizes and evaluating compression efficiency. By employing fundamental programming principles like modular design and data structures, it ensures efficiency and clarity in file processing. Through intuitive user prompts and error handling, the program enhances usability for individuals of all skill levels. The utilization of linked lists and stacks enables seamless organization and presentation of file data, facilitating smooth interaction with users. With its ability to calculate compression ratios, the program provides valuable insights into the effectiveness of compression techniques, empowering users to make informed decisions about file management strategies. Overall, the program serves as a valuable tool for developers and users alike, facilitating effective file management and compression analysis within a programming environment.

**REFERENCES**

1. [GeeksforGeeks](https://www.geeksforgeeks.org/) - GeeksforGeeks is a well-known website with a vast collection of articles and tutorials on various programming topics, including data structures, algorithms, and file processing.

2. [Stack Overflow](https://stackoverflow.com/) - Stack Overflow is a popular question and answer platform where programmers can ask and answer questions on a wide range of programming topics.

3. [C Programming: A Modern Approach](https://knking.com/books/c2/) by K.N. King - This book provides comprehensive coverage of the C programming language, including data structures, file processing, and modular programming technique.

**APPENDIX**

new\_node->data.size\_byte = size\_byte;

new\_node->next = NULL;

return new\_node;

}

// Function to convert gigabytes to megabytes

double convert\_gb\_to\_mb(double size\_gb) {

return size\_gb \* GB\_TO\_MB;

}

// Function to convert gigabytes to bytes

double convert\_gb\_to\_byte(double size\_gb) {

return size\_gb \* GB\_TO\_BYTE;

}

// Function to calculate the compression ratio #include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define GB\_TO\_MB 1024

#define GB\_TO\_BYTE (1024 \* 1024 \* 1024)

// Define a struct to hold file sizes in GB, MB, and bytes, and file name

typedef struct {

char file\_name[100];

double size\_gb;

double size\_mb;

double size\_byte;

} FileSize;

// Define a node in the singly linked list

typedef struct ListNode {

FileSize data;

struct ListNode\* next;

} ListNode;

// Define the Stack

typedef struct {

ListNode\* top;

} Stack;

// Function to create a new node

ListNode\* create\_node(char\* file\_name, double size\_gb, double size\_mb, double size\_byte) {

ListNode\* new\_node = (ListNode\*)malloc(sizeof(ListNode));

strcpy(new\_node->data.file\_name, file\_name);

new\_node->data.size\_gb = size\_gb;

new\_node->data.size\_mb = size\_mb;

double calculate\_compression\_ratio(double original\_size, double compressed\_size) {

return original\_size / compressed\_size;

}

// Stack operations

Stack\* create\_stack() {

Stack\* s = (Stack\*)malloc(sizeof(Stack));

s->top = NULL;

return s;

}

void push(Stack\* s, char\* file\_name, double size\_gb, double size\_mb, double size\_byte) {

ListNode\* temp = create\_node(file\_name, size\_gb, size\_mb, size\_byte);

temp->next = s->top;

s->top = temp;

}

ListNode\* pop(Stack\* s) {

if (s->top == NULL) return NULL;

ListNode\* temp = s->top;

s->top = s->top->next;

return temp;

}

// Function to print and free the stack

void print\_and\_free\_stack(Stack\* s) {

ListNode\* node;

while ((node = pop(s)) != NULL) {

printf("File: %s, Size: %.2f GB (%.2f MB, %.2f Bytes)\n", node->data.file\_name, node->data.size\_gb, node->data.size\_mb, node->data.size\_byte);

free(node);

}

free(s);

}

int main() {

char file\_name[100];

double original\_size\_gb, compressed\_size;

int choice;

// Get the file name from the user

printf("Enter a file name: ");

scanf("%s", file\_name);

// Get the original file size in GB from the user

printf("Enter the original size of the file in GB: ");

scanf("%lf", &original\_size\_gb);

// Get the user's choice for conversion

printf("Enter 1 to convert to Bytes or 2 to convert to MB: ");

scanf("%d", &choice);

if (choice == 1) {

// Convert to bytes

compressed\_size = convert\_gb\_to\_byte(original\_size\_gb);

printf("Enter the compressed file size in Bytes: ");

} else if (choice == 2) {

// Convert to megabytes

compressed\_size = convert\_gb\_to\_mb(original\_size\_gb);

printf("Enter the compressed file size in MB: ");

} else {

printf("Invalid choice.\n");

return 1;

}

scanf("%lf", &compressed\_size);

// Create a stack

Stack\* s = create\_stack();

// Add the original and compressed file sizes to the stack

double size\_mb = convert\_gb\_to\_mb(original\_size\_gb);

double size\_byte = convert\_gb\_to\_byte(original\_size\_gb);

push(s, file\_name, original\_size\_gb, size\_mb, size\_byte);

if (choice == 1) {

push(s, "Compressed", 0.0, 0.0, compressed\_size);

} else if (choice == 2) {

push(s, "Compressed", 0.0, compressed\_size, 0.0);

}

// Calculate the compression ratio

double compression\_ratio;

if (choice == 1) {

compression\_ratio = calculate\_compression\_ratio(size\_byte, compressed\_size);

} else if (choice == 2) {

compression\_ratio = calculate\_compression\_ratio(size\_mb, compressed\_size);

}

// Print the stack

printf("File Sizes in Stack (LIFO):\n");

print\_and\_free\_stack(s);

// Print the compression ratio

printf("Compression Ratio: %.2f\n", compression\_ratio);

return 0;

}