**Projects**

1. Student Grade Management System:

Program: The program should allow users to manage student grades. This includes adding, deleting, and modifying student data and calculation of grades.

Dynamic Arrays: The program should use dynamic arrays to store and manipulate student records. This will allow the program to handle any number of students.

Functions: The program should use functions to perform operations on student data. This will make the program more modular and easier to maintain.

File Handling: The program should be able to read and write student records to a file. This will allow the program to save and load data, and to share data with other programs.

Program Flow: Students list should be available as a file with their roll numbers, names and GPA. Their can be more than one files to manage different batches. The user should be asked which batch he wants to open. The user will be provided with the options to edit the file information or calculate grade of a particular course. Upon selecting a course, that particular course file contents will be displayed. It will have all the detailed marks of assignments, quizzes, class participations, mid term and final term. User can select to update or delete marks for any particular assessment. He can also update marks of any particular student. User can also select the option of calculating the final grade. For that he will be asked to enter the weightage of each assessment (For example final term : 50%, midterm: 20%, CP: 5%, assignments: 10%, quizzes: 15%). Then final grade of all students will be calculated and stored in the course file.

1. Tic-Tac-Toe Game:

Game: The game should be a command-line version of the popular Tic-Tac-Toe game.

2D Array: The game should use a 2D array to represent the game board. This will allow the game to keep track of the state of the game board.

Functions: The game should use functions to handle player moves, check for a winner, and display the game board. This will make the code more modular and easier to maintain.

Pointers: The game should use pointers to manage the game state and memory dynamically. This will allow the game to handle any size game board.

1. Phonebook Application:

Program: The program should be a command-line phonebook application.

Dynamic Arrays: The program should use dynamic arrays of character arrays (strings) to store contact information. This will allow the program to handle any number of contacts.

Functions: The program should use functions to handle different operations on the phonebook, such as adding, searching, modifying, and deleting contacts. This will make the program more modular and easier to maintain.

File Handling: The program should be able to read and write the phonebook to a file. This will allow the program to save and load the phonebook, and to share the phonebook with other programs.

1. Matrix Operations:

Program: The program should be a command-line matrix operations program.

2D Arrays: The program should use 2D arrays to store and manipulate matrix elements. This will allow the program to handle any size matrix.

Functions: The program should use functions to perform the matrix operations, such as addition, subtraction, multiplication, and transposition. This will make the program more modular and easier to maintain.

File Handling: The program should be able to read and write matrices to a file. This will allow the program to save and load matrices, and to share matrices with other programs.

1. Online Shopping Cart:

Program: The program should be a command-line online shopping cart program.

Dynamic Arrays: The program should use dynamic arrays to store the products added by the user. This will allow the program to handle any number of products.

Functions: The program should use functions to add items to the cart, remove items, and calculate the total cost. This will make the program more modular and easier to maintain.

File Handling: The program should be able to load product information from a file (e.g., name, price) and save the cart contents. This will allow the program to save and load the cart, and to share the cart with other programs.

Flow of the program: Load the product information (name/price/items in stock) from a file. Ask the user which items he wants to add in his cart and quantity of the item. Keep asking the user until he enters ‘E’ (for exit). Show him the total bill and ask him to confirm his order. If the user confirms the order, subtract the number of items from the file accordingly and ask for all his details for confirming order. Also, save the order of the user alongwith all his details in a file with his name.