Group - 1

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Customer {
    int accountNumber;
    char name[100];
    float balance;
};

void createAccount(struct Customer *customers, int *count) {
    if (*count >= 100) {
        printf("Customer database is full. Cannot create more accounts.\n");
        return;
    }

    struct Customer newCustomer;

    printf("Enter customer details:\n");
    printf("Account Number: ");
    scanf("%d", &newCustomer.accountNumber);
    printf("Customer Name: ");
    scanf(" %[^\n]", newCustomer.name);
    printf("Initial Balance: ");
    scanf("%f", &newCustomer.balance);

    customers[*count] = newCustomer;
    (*count)++;

    printf("Account created successfully.\n");
}

void deposit(struct Customer *customers, int count) {
    int accountNumber;
    float amount;
    int found = 0;

    printf("Enter the account number: ");
    scanf("%d", &accountNumber);

    for (int i = 0; i < count; i++) {
        if (customers[i].accountNumber == accountNumber) {
            printf("Enter the amount to deposit: ");
            scanf("%f", &amount);

            customers[i].balance += amount;
            printf("Amount deposited successfully.\n");
            printf("Updated balance: %.2f\n", customers[i].balance);

            found = 1;
            break;
        }
    }

    if (!found) {
        printf("Account not found.\n");
    }
}

void withdraw(struct Customer *customers, int count) {
    int accountNumber;
    float amount;
    int found = 0;

    printf("Enter the account number: ");
    scanf("%d", &accountNumber);

    for (int i = 0; i < count; i++) {
        if (customers[i].accountNumber == accountNumber) {
            printf("Enter the amount to withdraw: ");
            scanf("%f", &amount);

            if (amount > customers[i].balance) {
                printf("Insufficient balance.\n");
            } else {
```

```
                customers[i].balance -= amount;
                printf("Amount withdrawn successfully.\n");
                printf("Updated balance: %.2f\n", customers[i].balance);
            }

            found = 1;
            break;
        }
    }

    if (!found) {
        printf("Account not found.\n");
    }
}

void viewBalance(struct Customer *customers, int count) {
    int accountNumber;
    int found = 0;

    printf("Enter the account number: ");
    scanf("%d", &accountNumber);

    for (int i = 0; i < count; i++) {
        if (customers[i].accountNumber == accountNumber) {
            printf("Account Number: %d\n", customers[i].accountNumber);
            printf("Customer Name: %s\n", customers[i].name);
            printf("Current Balance: %.2f\n", customers[i].balance);

            found = 1;
            break;
        }
    }

    if (!found) {
        printf("Account not found.\n");
    }
}

int main() {
    struct Customer customers[100];
    int count = 0;
    int choice;

    printf("Bank Management System\n");

    while (1) {
        printf("\nSelect an option:\n");
        printf("1. Create an account\n");
        printf("2. Deposit\n");
        printf("3. Withdraw\n");
        printf("4. View balance\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch(choice) {
            case 1:
                createAccount(customers, &count);
                break;
            case 2:
                deposit(customers, count);
                break;
            case 3:
                withdraw(customers, count);
                break;
            case 4:
                viewBalance(customers, count);
                break;
            case 5:
                printf("Thank you for using the bank management system.\n");
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }
```

```
    return 0;
}
```

Group-2

```c
#include <stdio.h>

char board[3][3]; // Tic Tac Toe board

void initializeBoard() {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            board[i][j] = ' ';
        }
    }
}

void printBoard() {
    printf("\n");
    printf(" %c | %c | %c\n", board[0][0], board[0][1], board[0][2]);
    printf("---+---+---\n");
    printf(" %c | %c | %c\n", board[1][0], board[1][1], board[1][2]);
    printf("---+---+---\n");
    printf(" %c | %c | %c\n", board[2][0], board[2][1], board[2][2]);
    printf("\n");
}

int checkWin() {
    // Check rows
    for (int i = 0; i < 3; i++) {
        if (board[i][0] == board[i][1] && board[i][1] == board[i][2] && board[i][0] != ' ') {
            return 1;
        }
    }

    // Check columns
    for (int i = 0; i < 3; i++) {
        if (board[0][i] == board[1][i] && board[1][i] == board[2][i] && board[0][i] != ' ') {
            return 1;
        }
    }

    // Check diagonals
    if ((board[0][0] == board[1][1] && board[1][1] == board[2][2] && board[0][0] != ' ') ||
        (board[0][2] == board[1][1] && board[1][1] == board[2][0] && board[0][2] != ' ')) {
        return 1;
    }

    // Check for a tie
    int tie = 1;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (board[i][j] == ' ') {
                tie = 0;
                break;
            }
        }
    }
    if (tie) {
        return 2;
    }

    return 0;
}

int main() {
    int currentPlayer = 1; // Player 1 starts
    int row, col;
    int gameOver = 0;
    int winner;

    initializeBoard();

    printf("Tic Tac Toe Game\n");

    while (!gameOver) {
        printBoard();

        printf("Player %d's turn.\n", currentPlayer);
        printf("Enter the row (0-2): ");
```

```
        scanf("%d", &row);
        printf("Enter the column (0-2): ");
        scanf("%d", &col);

        if (row < 0 || row > 2 || col < 0 || col > 2 || board[row][col] != ' ') {
            printf("Invalid move. Please try again.\n");
            continue;
        }

        if (currentPlayer == 1) {
      board[row][col] = 'X';
      currentPlayer = 2; // switch to player 2
  }
 else {
     board[row][col] = 'O';
     currentPlayer = 1; // switch to player 1
 }
}
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_STUDENTS 50
typedef struct
{
char name[50];
int roll_number;
int marks;
char grade;
} student;
int main()
{
student students[MAX_STUDENTS];
int num_students = 0;
int choice = 0;
int i = 0;
while (1)
{
printf("1. Add student\n");
printf("2. View all students\n");
printf("3. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
switch (choice)
{
case 1:
if (num_students == MAX_STUDENTS)
printf("Maximum number of students reached.\n");
else
{
printf("Enter student name: ");
scanf("%s", students[num_students].name);
printf("Enter roll number: ");
scanf("%d", &students[num_students].roll_number);
printf("Enter marks: ");
scanf("%d", &students[num_students].marks);
students[num_students].grade =((students[num_students].marks >= 90)?'A':(students[num_students].marks >= 80)?'B':(students[num_students].marks >= 70)?'C':(students[num_students].marks
num_students++;
}
break;
case 2:
printf("Name\t\tRoll Number\tMarks\tGrade\n");
for (i = 0; i < num_students; i++)
printf("%s\t%d\t%d\t%c\n", students[i].name, students[i].roll_number, students[i].marks, students[i].grade);
break;
case 3:
exit(0);
default:
printf("Invalid choice. Please try again.\n");
break;
}
}
return 0;
}
```

Group - 3

Group - 4

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_TRANSLATIONS 100

// Structure to store translation pair
typedef struct {
    char source[100];
    char target[100];
} Translation;

// Global array to store translations
Translation translations[MAX_TRANSLATIONS];
int num_translations = 0;

// Function Prototypes
void addTranslation();
char* translateText(const char* text);

int main() {
    int choice;
    char text[100];

    do {
        printf("\nLanguage Translator\n");
        printf("1. Add Translation\n");
        printf("2. Translate Text\n");
        printf("0. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addTranslation();
                break;
            case 2:
                printf("Enter text to translate: ");
                scanf(" %[^\n]s", text);
                printf("Translated Text: %s\n", translateText(text));
                break;
            case 0:
                printf("Exiting program. Goodbye!\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
                break;
        }
    } while (choice != 0);

    return 0;
}

// Function to add a translation pair to the translator
void addTranslation() {
    if (num_translations == MAX_TRANSLATIONS) {
        printf("Cannot add more translations. Limit reached.\n");
        return;
    }

    printf("Enter source language: ");
    scanf(" %[^\n]s", translations[num_translations].source);
    printf("Enter target language: ");
    scanf(" %[^\n]s", translations[num_translations].target);

    printf("Translation added successfully.\n");
    num_translations++;
}

// Function to translate text based on available translations
char* translateText(const char* text) {
    char* translated_text = (char*)malloc(1000 * sizeof(char));
    translated_text[0] = '\0';

    for (int i = 0; i < num_translations; i++) {
```

```
        if (strcmp(translations[i].source, text) == 0) {
            strcpy(translated_text, translations[i].target);
            break;
        }
    }

    if (translated_text[0] == '\0') {
        strcpy(translated_text, "Translation not found.");
    }

    return translated_text;
}
```

```c
#include <stdio.h>

int main() {
    int pin = 1234;
    int balance = 5000;
    int option, enteredPin, withdrawAmount, depositAmount;

    printf("Welcome to the ATM Simulator!\n");

    while (1) {
        printf("\nPlease enter your PIN: ");
        scanf("%d", &enteredPin);

        if (enteredPin != pin) {
            printf("Invalid PIN. Please try again.\n");
            continue;
        }

        printf("\nATM Menu:\n");
        printf("1. Check Balance\n");
        printf("2. Withdraw\n");
        printf("3. Deposit\n");
        printf("4. Exit\n");
        printf("Enter your option: ");
        scanf("%d", &option);

        switch (option) {
            case 1:
                printf("Your current balance is: $%d\n", balance);
                break;
            case 2:
                printf("Enter the amount to withdraw: ");
                scanf("%d", &withdrawAmount);
                if (withdrawAmount > balance) {
                    printf("Insufficient balance.\n");
                } else {
                    balance -= withdrawAmount;
                    printf("Withdrawal successful. Remaining balance is: $%d\n", balance);
                }
                break;
            case 3:
                printf("Enter the amount to deposit: ");
                scanf("%d", &depositAmount);
                balance += depositAmount;
                printf("Deposit successful. Updated balance is: $%d\n", balance);
                break;
            case 4:
                printf("Thank you for using the ATM. Goodbye!\n");
                return 0;
            default:
                printf("Invalid option. Please try again.\n");
                break;
        }
    }

    return 0;
}
```

Group - 5

Group 6

```c
#include <stdio.h>
#include <math.h>

int main() {
    int option;
    double num, result;

    printf("Scientific Calculator\n");

    while (1) {
        printf("\nSelect an operation:\n");
        printf("1. Square Root\n");
        printf("2. Exponentiation\n");
        printf("3. Logarithm (base 10)\n");
        printf("4. Natural Logarithm (base e)\n");
        printf("5. Sine\n");
        printf("6. Cosine\n");
        printf("7. Tangent\n");
        printf("8. Exit\n");
        printf("Enter your option: ");
        scanf("%d", &option);

        if (option == 8) {
            printf("Thank you for using the calculator. Goodbye!\n");
            break;
        }

        printf("Enter a number: ");
        scanf("%lf", &num);

        switch (option) {
            case 1:
                result = sqrt(num);
                printf("Square Root: %lf\n", result);
                break;
            case 2:
                result = pow(num, 2);
                printf("Exponentiation: %lf\n", result);
                break;
            case 3:
                result = log10(num);
                printf("Logarithm (base 10): %lf\n", result);
                break;
            case 4:
                result = log(num);
                printf("Natural Logarithm (base e): %lf\n", result);
                break;
            case 5:
                result = sin(num);
                printf("Sine: %lf\n", result);
                break;
            case 6:
                result = cos(num);
                printf("Cosine: %lf\n", result);
                break;
            case 7:
                result = tan(num);
                printf("Tangent: %lf\n", result);
                break;
            default:
                printf("Invalid option. Please try again.\n");
                break;
        }
    }

    return 0;
}
```

Group - 7

```c
#include <stdio.h>
#include <stdlib.h>

void createFile() {
    FILE *file;
    char fileName[100];

    printf("Enter the name of the file: ");
    scanf("%s", fileName);

    file = fopen(fileName, "w");

    if (file == NULL) {
        printf("Error creating file.\n");
        return;
    }

    printf("File created successfully.\n");

    fclose(file);
}

void writeFile() {
    FILE *file;
    char fileName[100];
    char content[1000];

    printf("Enter the name of the file: ");
    scanf("%s", fileName);

    file = fopen(fileName, "a");

    if (file == NULL) {
        printf("Error opening file.\n");
        return;
    }

    printf("Enter the content to write (max 1000 characters):\n");
    scanf(" %[^\n]", content);

    fprintf(file, "%s\n", content);
    printf("Content written to the file.\n");

    fclose(file);
}

void readFile() {
    FILE *file;
    char fileName[100];
    char content[1000];

    printf("Enter the name of the file: ");
    scanf("%s", fileName);

    file = fopen(fileName, "r");

    if (file == NULL) {
        printf("Error opening file.\n");
        return;
    }

    printf("File content:\n");

    while (fgets(content, sizeof(content), file) != NULL) {
        printf("%s", content);
    }

    fclose(file);
}

void deleteFile() {
    char fileName[100];

    printf("Enter the name of the file: ");
    scanf("%s", fileName);
```

```c
    if (remove(fileName) == 0) {
        printf("File deleted successfully.\n");
    } else {
        printf("Error deleting file.\n");
    }
}

int main() {
    int choice;

    printf("File Management System\n");

    while (1) {
        printf("\nSelect an option:\n");
        printf("1. Create a File\n");
        printf("2. Write to a File\n");
        printf("3. Read a File\n");
        printf("4. Delete a File\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                createFile();
                break;
            case 2:
                writeFile();
                break;
            case 3:
                readFile();
                break;
            case 4:
                deleteFile();
                break;
            case 5:
                printf("Thank you for using the file management system. Goodbye!\n");
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
                break;
        }
    }

    return 0;
}
```

Group-8

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Product {
    int id;
    char name[100];
    int quantity;
    float price;
};

void addProduct(struct Product *inventory, int *count) {
    if (*count >= 100) {
        printf("Inventory is full. Cannot add more products.\n");
        return;
    }

    struct Product newProduct;

    printf("Enter product details:\n");
    printf("Product ID: ");
    scanf("%d", &newProduct.id);
    printf("Product Name: ");
    scanf(" %[^\n]", newProduct.name);
    printf("Quantity: ");
    scanf("%d", &newProduct.quantity);
    printf("Price: ");
    scanf("%f", &newProduct.price);

    inventory[*count] = newProduct;
    (*count)++;

    printf("Product added successfully.\n");
}

void updateProduct(struct Product *inventory, int count) {
    int productId;
    int found = 0;

    printf("Enter the product ID to update: ");
    scanf("%d", &productId);

    for (int i = 0; i < count; i++) {
        if (inventory[i].id == productId) {
            printf("Enter new details for the product:\n");
            printf("Product Name: ");
            scanf(" %[^\n]", inventory[i].name);
            printf("Quantity: ");
            scanf("%d", &inventory[i].quantity);
            printf("Price: ");
            scanf("%f", &inventory[i].price);

            printf("Product details updated successfully.\n");
            found = 1;
            break;
        }
    }

    if (!found) {
        printf("Product not found.\n");
    }
}

void deleteProduct(struct Product *inventory, int *count) {
    int productId;
    int found = 0;

    printf("Enter the product ID to delete: ");
    scanf("%d", &productId);

    for (int i = 0; i < *count; i++) {
        if (inventory[i].id == productId) {
            for (int j = i; j < *count - 1; j++) {
                inventory[j] = inventory[j + 1];
            }

            (*count)--;
```

```c
            printf("Product deleted successfully.\n");
            found = 1;
            break;
        }
    }

    if (!found) {
        printf("Product not found.\n");
    }
}

void displayInventory(struct Product *inventory, int count) {
    if (count == 0) {
        printf("Inventory is empty.\n");
        return;
    }

    printf("Product Inventory:\n");
    printf("ID\tName\tQuantity\tPrice\n");
    for (int i = 0; i < count; i++) {
        printf("%d\t%s\t%d\t\t%.2f\n", inventory[i].id, inventory[i].name, inventory[i].quantity, inventory[i].price);
    }
}

int main() {
    struct Product inventory[100];
    int count = 0;
    int choice;

    printf("Inventory Management System\n");

    while (1) {
        printf("\nSelect an option:\n");
        printf("1. Add a product\n");
        printf("2. Update a product\n");
        printf("3. Delete a product\n");
        printf("4. Display inventory\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addProduct(inventory, &count);
                break;

                        case 2:
                updateProduct(inventory, count);
                break;

            case 3:
                deleteProduct(inventory, &count);
                break;

            case 4:
                displayInventory(inventory, count);
                break;

            case 5:
                printf("Exiting the program...\n");
                exit(0);

            default:
                printf("Invalid choice. Please try again.\n");
                break;
        }
    }

    return 0;
}
```

Group-9

```c
#include <stdio.h>
#include <stdlib.h>
#include <curl/curl.h>
#include <cJSON.h>

// Function to handle CURL write callback
size_t write_callback(char *data, size_t size, size_t nmemb, char *buffer) {
    size_t total_size = size * nmemb;
    buffer = realloc(buffer, total_size + 1);
    if(buffer == NULL) {
        printf("Error: Unable to allocate memory.\n");
        return 0;
    }
    strncat(buffer, data, total_size);
    return total_size;
}

int main() {
    CURL *curl;
    CURLcode res;

    char *weather_api_url = "API_URL_HERE";
    char *api_key = "YOUR_API_KEY_HERE";

    char *buffer = malloc(4096 * sizeof(char));
    if(buffer == NULL) {
        printf("Error: Unable to allocate memory.\n");
        return 1;
    }
    buffer[0] = '\0';

    curl = curl_easy_init();
    if(curl) {
        char request_url[512];
        sprintf(request_url, "%s?apikey=%s", weather_api_url, api_key);

        curl_easy_setopt(curl, CURLOPT_URL, request_url);
        curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, write_callback);
        curl_easy_setopt(curl, CURLOPT_WRITEDATA, buffer);

        res = curl_easy_perform(curl);
        if(res != CURLE_OK) {
            printf("Error: %s\n", curl_easy_strerror(res));
            return 1;
        }

        curl_easy_cleanup(curl);

        // Parse JSON response
        cJSON *json = cJSON_Parse(buffer);
        if(json == NULL) {
            printf("Error: Failed to parse JSON.\n");
            free(buffer);
            return 1;
        }

        // Extract desired weather data from JSON
        cJSON *temperature = cJSON_GetObjectItem(json, "temperature");
        cJSON *humidity = cJSON_GetObjectItem(json, "humidity");
        cJSON *description = cJSON_GetObjectItem(json, "description");

        // Display weather data
        printf("Temperature: %.2fÂ°C\n", temperature->valuedouble);
        printf("Humidity: %.2f%%\n", humidity->valuedouble);
        printf("Description: %s\n", description->valuestring);

        // Clean up
        cJSON_Delete(json);
        free(buffer);
    }

    return 0;
}
```

Group-10

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Employee {
    int id;
    char name[100];
    int age;
    float salary;
};

void addEmployee(struct Employee *employees, int *count) {
    if (*count >= 100) {
        printf("Employee database is full. Cannot add more employees.\n");
        return;
    }

    struct Employee newEmployee;

    printf("Enter employee details:\n");
    printf("Employee ID: ");
    scanf("%d", &newEmployee.id);
    printf("Employee Name: ");
    scanf(" %[^\n]", newEmployee.name);
    printf("Employee Age: ");
    scanf("%d", &newEmployee.age);
    printf("Employee Salary: ");
    scanf("%f", &newEmployee.salary);

    employees[*count] = newEmployee;
    (*count)++;

    printf("Employee added successfully.\n");
}

void updateEmployee(struct Employee *employees, int count) {
    int employeeId;
    int found = 0;

    printf("Enter the employee ID to update: ");
    scanf("%d", &employeeId);

    for (int i = 0; i < count; i++) {
        if (employees[i].id == employeeId) {
            printf("Enter new details for the employee:\n");
            printf("Employee Name: ");
            scanf(" %[^\n]", employees[i].name);
            printf("Employee Age: ");
            scanf("%d", &employees[i].age);
            printf("Employee Salary: ");
            scanf("%f", &employees[i].salary);

            printf("Employee details updated successfully.\n");
            found = 1;
            break;
        }
    }

    if (!found) {
        printf("Employee not found.\n");
    }
}

void deleteEmployee(struct Employee *employees, int *count) {
    int employeeId;
    int found = 0;

    printf("Enter the employee ID to delete: ");
    scanf("%d", &employeeId);

    for (int i = 0; i < *count; i++) {
        if (employees[i].id == employeeId) {
            for (int j = i; j < *count - 1; j++) {
                employees[j] = employees[j + 1];
            }
```

```c
            (*count)--;
            printf("Employee deleted successfully.\n");
            found = 1;
            break;
        }
    }

    if (!found) {
        printf("Employee not found.\n");
    }
}

void displayEmployees(struct Employee *employees, int count) {
    if (count == 0) {
        printf("Employee database is empty.\n");
        return;
    }

    printf("Employee Database:\n");
    printf("ID\tName\tAge\tSalary\n");
    for (int i = 0; i < count; i++) {
        printf("%d\t%s\t%d\t%.2f\n", employees[i].id, employees[i].name, employees[i].age, employees[i].salary);
    }
}

int main() {
    struct Employee employees[100];
    int count = 0;
    int choice;

    printf("Employee Management System\n");

    while (1) {
        printf("\nSelect an option:\n");
        printf("1. Add an employee\n");
        printf("2. Update an employee\n");
        printf("3. Delete an employee\n");
        printf("4. Display employees\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addEmployee(employees, &count);
                break;
            case 2:
                updateEmployee(employees, count);
                break;
            case 3:
                deleteEmployee(employees, &count);
                break;
            case 4:
                displayEmployees(employees, count);
                break;
            case 5:
                printf("Exiting Employee Management System. Goodbye!\n");
                exit(0);
            default:
                printf("Invalid choice. Please select a valid option.\n");
        }
    }

    return 0;
}
```

1) In ~~g~~ the project you need the <curl-h> directory for it to work

2) If using MinGW you should give the commands gcc (name of the project.c) then just write a

[Note* open CMD in the folder where the projects are stored]

3) It's good to have ego, till you have the facts to back it up.