**Assignment #02 Roll No. 242721 Muhammad Aezaz Hasan**

**Week 1.**

**Case 01.**

**Week 2.**

**Case 01.**

#include <iostream>

using namespace std;

int main()

{

    int a;

    int b;

    char operation;

    cout << "Enter the first number : ";

    cin >> a;

    cout << "Enter the Second number : ";

    cin >> b;

    cout << "Select the operation (+,-,\*,/) : ";

    cin >> operation;

    switch (operation)

    {

    case '+':

        cout << "The addition of " << a << " and " << b << " is : " << a + b << endl;

        break;

    case '-':

        cout << "The subtraction of " << a << " and " << b << " is : " << a - b << endl;

        break;

    case '\*':

        cout << "The Multiplication of " << a << " and " << b << " is : " << a \* b << endl;

        break;

    case '/':

        cout << "The division of " << a << " and " << b << " is : " << a / b << endl;

        break;

    }

    return 0;

}

**Case 02.**

#include <iostream>

using namespace std;

int main()

{

    float totalcost = 0.00;

    int choice;

    do

    {

        cout << "Kindly select from the Menu listed : " << endl;

        cout << "1. Soup $2.5\n2. Shampoo $3.00\n3. Vegetable $1.50\n4. Oil $5.00\n5. Handwash $2.00\n6. Exit menu and print receipt\n";

        cin >> choice;

        switch (choice)

        {

        case 1:

            totalcost += 2.50;

            break;

        case 2:

            totalcost += 3.00;

            break;

        case 3:

            totalcost += 1.50;

            break;

        case 4:

            totalcost += 5.00;

            break;

        case 5:

            totalcost += 2.00;

            break;

        case 6:

            cout << "exiting menu and printing total bill.." << endl;

            break;

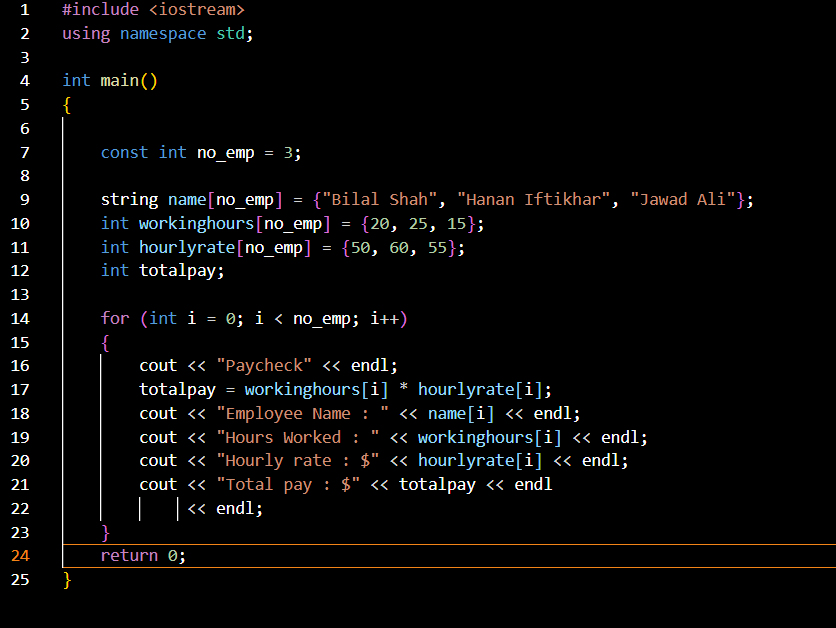
        }

    } while (choice != 6);

    cout << "The total bill of the purchased items is : $" << totalcost;

}

**Case 03.**



**Case 04.**

#include <iostream>

#include <cctype>

using namespace std;

bool checkemail(string &email)

{

    bool attherate = false;

    bool dot = false;

    for (int i = 0; i < email.length(); i++)

    {

        if (email[i] == '@')

        {

            attherate = true;

        }

        if (email[i] == '.')

        {

            if (attherate)

                dot = true;

        }

    }

    return attherate && dot;

}

bool checkusername(string &username)

{

    if (username.length() < 3 && username.length() > 15)

    {

        return false;

    }

    for (int i = 0; i < username.length(); i++)

    {

        if (!isalnum(username[i]))

        {

            return false;

        }

    }

    return true;

};

bool checkpassword(string &password)

{

    if (password.length() < 8)

    {

        return false;

    }

    bool letter = false;

    bool digit = false;

    for (int i = 0; i < password.length(); i++)

    {

        if (isalpha(password[i]))

        {

            letter = true;

        }

        if (isdigit(password[i]))

        {

            digit = true;

        }

    }

    return letter && digit;

};

int main()

{

    string email;

    string username;

    string password;

    cout << "Enter your email : ";

    getline(cin, email);

    if (!checkemail(email))

    {

        cout << "Invalid email" << endl;

        return 1;

        ;

    }

    cout << "Enter Username (3-15 characters) : ";

    getline(cin, username);

    if (!checkusername(username))

    {

        cout << "Invalid username" << endl;

        return 1;

    }

    cout << "Enter password : ";

    getline(cin, password);

    if (!checkpassword(password))

    {

        cout << "Invalid password" << endl;

        return 1;

    }

    cout << "Registered Successfully!" << endl;

}

**Week 3.**

**Case 01.**

#include <iostream>

using namespace std;

const int lanes = 4;

int greenlight(int vehcount[])

{

    int maxveh = 0;

    int lanewithmax = 0;

    for (int i = 0; i < lanes; i++)

    {

        if (vehcount[i] > maxveh)

        {

            maxveh = vehcount[i];

            lanewithmax = i;

        }

    }

    return lanewithmax;

}

void signal(int vehcount[])

{

    int lane = greenlight(vehcount);

    for (int i = 0; i < lanes; i++)

    {

        if (i == lane)

        {

            cout << "Lane " << i + 1 << " has Green light (" << vehcount[i] << ")" << endl;

        }

        else

        {

            cout << "Lane " << i + 1 << " has Red light (" << vehcount[i] << ")" << endl;

        }

    }

}

int main()

{

    int vehcount[lanes] = {20, 30, 14, 25};

    cout << "Current Traffic Signals status : " << endl;

    signal(vehcount);

}

**Case 02.**

#include <iostream>

using namespace std;

int discount(float &total, int count)

{

    int discount;

    if (count >= 2)

    {

        discount = total \* 20 / 100;

        total = (0.00 + total) - discount;

        return total;

    }

    return total;

}

int main()

{

    int choice;

    float total = 0;

    int count = 0;

    do

    {

        count++;

        cout << "Menu \n20% Discount if you buy 2 or more than 2 items" << endl;

        cout << "1. Burger $5.00 \n2. Pizza $7.00\n3. Steak $10.00 \n4. Sushi $4.00 \n5. Combo Platter $12.00\n6. Exit Menu." << endl;

        cin >> choice;

        switch (choice)

        {

        case 1:

            cout << "Burger added" << endl;

            total = total + 5.00;

            discount(total, count);

            break;

        case 2:

            cout << "Pizza added" << endl;

            total = total + 7.00;

            discount(total, count);

            break;

        case 3:

            cout << "Steak added" << endl;

            total = total + 10.00;

            discount(total, count);

            break;

        case 4:

            cout << "Sushi added" << endl;

            total = total + 4.00;

            discount(total, count);

            break;

        case 5:

            cout << "Combo Platter added" << endl;

            total = total + 12.00;

            discount(total, count);

            break;

        case 6:

            cout << "Exiting menu... printing receipt." << endl;

            break;

        }

    } while (choice != 6);

    if (count >= 3)

    {

        cout << "Your total Bill after discount is : $" << total << endl;

    }

    else

    {

        cout << "Your total Bill is : $" << total << endl;

    }

}

**Case 03.**

#include <iostream>

#include <string>

using namespace std;

void mainRoom() {

    string command;

    cout << "You are in a dark room. There are two doors: one to the north and one to the east." << endl;

    cout << "What do you want to do? (Type 'go north', 'go east', or 'look around')\n";

    getline(cin, command);

    if (command == "go north") {

        cout << "You go north and find yourself in a mysterious forest.\n";

    }

    else if (command == "go east") {

        cout << "You go east and find yourself in a library full of ancient books.\n";

    }

    else if (command == "look around") {

        cout << "You see two doors: one to the north and one to the east.\n";

    }

    else {

        cout << "Invalid command. Try again.\n";

    }

}

void forestRoom() {

    string command;

    cout << "You are in a dark, eerie forest. There is a path leading deeper into the woods and a small hut nearby." << endl;

    cout << "What do you want to do? (Type 'go deeper', 'enter hut', or 'look around')\n";

    getline(cin, command);

    if (command == "go deeper") {

        cout << "You venture deeper into the woods, but the forest becomes more ominous.\n";

    }

    else if (command == "enter hut") {

        cout << "You enter the hut and find an old man who offers you a magical potion.\n";

    }

    else if (command == "look around") {

        cout << "You see the path leading deeper into the woods and the hut nearby.\n";

    }

    else {

        cout << "Invalid command. Try again.\n";

    }

}

void libraryRoom() {

    string command;

    cout << "You are in a dusty library with shelves full of old, mysterious books. There is a desk with a glowing book." << endl;

    cout << "What do you want to do? (Type 'read book', 'take book', or 'look around')\n";

    getline(cin, command);

    if (command == "read book") {

        cout << "You open the glowing book and discover a secret passage!\n";

    }

    else if (command == "take book") {

        cout << "You take the book from the desk, and it suddenly starts to glow brighter.\n";

    }

    else if (command == "look around") {

        cout << "You see shelves full of books and the glowing book on the desk.\n";

    }

    else {

        cout << "Invalid command. Try again.\n";

    }

}

void startGame() {

    string command;

    cout << "Welcome to the Adventure Game!\n";

    mainRoom();

    while (true) {

        cout << "\nWhere would you like to go next? (Type 'go north' to the forest or 'go east' to the library)\n";

        getline(cin, command);

        if (command == "go north") {

            forestRoom();

        }

        else if (command == "go east") {

            libraryRoom();

        }

        else {

            cout << "Invalid command. Try again.\n";

        }

        cout << "\nDo you want to keep playing? (yes/no)\n";

        getline(cin, command);

        if (command == "no") {

            cout << "Thanks for playing! Goodbye!\n";

            break;

        } else if (command == "yes") {

            cout << "Let's continue the adventure...\n";

        } else {

            cout << "Invalid command. Exiting the game...\n";

            break;

        }

    }

}

int main() {

    startGame();

    return 0;

}

**Week 4.**

**Case 01.**

#include <iostream>

#include <string>

#include <cstdlib>

#include <ctime>

using namespace std;

const int MAZE\_SIZE = 5;

int robotX = 0, robotY = 0;

int maze[MAZE\_SIZE][MAZE\_SIZE] = {

    {0, 1, 0, 0, 0},

    {0, 1, 0, 1, 0},

    {0, 0, 0, 1, 0},

    {0, 1, 0, 0, 0},

    {0, 0, 0, 1, 9}  // Goal at (4, 4)

};

void printMaze() {

    for (int i = 0; i < MAZE\_SIZE; i++) {

        for (int j = 0; j < MAZE\_SIZE; j++) {

            if (i == robotX && j == robotY)

                cout << "R ";  // Robot position

            else if (maze[i][j] == 1)

                cout << "# ";  // Wall

            else if (maze[i][j] == 9)

                cout << "G ";  // Goal

            else

                cout << ". ";  // Empty space

        }

        cout << endl;

    }

}

void moveUp() {

    if (robotX > 0 && maze[robotX - 1][robotY] != 1) {

        robotX--;

    } else {

        cout << "Can't move up, there's a wall!\n";

    }

}

void moveDown() {

    if (robotX < MAZE\_SIZE - 1 && maze[robotX + 1][robotY] != 1) {

        robotX++;

    } else {

        cout << "Can't move down, there's a wall!\n";

    }

}

void moveLeft() {

    if (robotY > 0 && maze[robotX][robotY - 1] != 1) {

        robotY--;

    } else {

        cout << "Can't move left, there's a wall!\n";

    }

}

void moveRight() {

    if (robotY < MAZE\_SIZE - 1 && maze[robotX][robotY + 1] != 1) {

        robotY++;

    } else {

        cout << "Can't move right, there's a wall!\n";

    }

}

bool hasReachedGoal() {

    return maze[robotX][robotY] == 9;

}

void simulateRobot() {

    string command;

    while (!hasReachedGoal()) {

        cout << "\nCurrent Maze Status: \n";

        printMaze();

        cout << "What should the robot do? (up/down/left/right): ";

        getline(cin, command);

        if (command == "up") {

            moveUp();

        } else if (command == "down") {

            moveDown();

        } else if (command == "left") {

            moveLeft();

        } else if (command == "right") {

            moveRight();

        } else {

            cout << "Invalid command. Please try again.\n";

        }

        if (hasReachedGoal()) {

            cout << "Congratulations! The robot has reached the goal!\n";

        }

    }

}

int main() {

    srand(time(0));

    cout << "Robot Simulation Start!\n";

    simulateRobot();

    return 0;

}

**Case 02.**

#include <iostream>

#include <string>

using namespace std;

void displaymenu() {

    cout << "Menu\n";

    cout << "1. Item A - $10\n";

    cout << "2. Item B - $15\n";

    cout << "3. Item C - $20\n";

}

void couponcodes() {

    cout << "Available coupon codes:\n";

    cout << "DISCOUNT10 - 10% off\n";

    cout << "SAVE20 - $20 off\n";

    cout << "FREESHIP - Free shipping\n";

}

bool validatecoupon(const string& coupon) {

    return coupon == "DISCOUNT10" || coupon == "SAVE20" || coupon == "FREESHIP";

}

int main() {

    int choice;

    string coupon;

    cout << "Welcome to Dwatson" << endl;

    do {

        cout << "Choose one" << endl;

        cout << "1. Display Menu\n2. Display Coupon Codes for Discounts\n3. Print Bill and exit store.." << endl;

        cin >> choice;

        switch (choice) {

            case 1:

                displaymenu();

                break;

            case 2:

                couponcodes();

                do {

                    cout << "Enter coupon code: ";

                    cin >> coupon;

                    if (!validatecoupon(coupon)) {

                        cout << "Invalid coupon code. Please try again.\n";

                    }

                } while (!validatecoupon(coupon));

                cout << "Coupon applied successfully: " << coupon << endl;

                break;

            case 3:

                cout << "Printing receipt and exiting store..." << endl;

                break;

            default:

                cout << "Invalid choice. Please try again." << endl;

        }

    } while (choice != 3);

    return 0;

}

**Case 03.**

#include <iostream>

#include <string>

using namespace std;

const int BOARD\_SIZE = 8;

char board[BOARD\_SIZE][BOARD\_SIZE];

void displayBoard() {

    cout << "  A B C D E F G H" << endl;

    for (int i = 0; i < BOARD\_SIZE; ++i) {

        cout << 8 - i << " ";

        for (int j = 0; j < BOARD\_SIZE; ++j) {

            cout << board[i][j] << " ";

        }

        cout << 8 - i << endl;

    }

    cout << "  A B C D E F G H" << endl;

}

void initializeBoard() {

    for (int i = 0; i < BOARD\_SIZE; ++i) {

        for (int j = 0; j < BOARD\_SIZE; ++j) {

            board[i][j] = '.';

        }

    }

    for (int i = 0; i < BOARD\_SIZE; ++i) {

        board[1][i] = 'P';

        board[6][i] = 'p';

    }

    string whitePieces = "RNBQKBNR";

    string blackPieces = "rnbqkbnr";

    for (int i = 0; i < BOARD\_SIZE; ++i) {

        board[0][i] = whitePieces[i];

        board[7][i] = blackPieces[i];

    }

}

bool isValidPosition(int row, int col) {

    return row >= 0 && row < BOARD\_SIZE && col >= 0 && col < BOARD\_SIZE;

}

bool checkWinCondition(char king) {

    for (int i = 0; i < BOARD\_SIZE; ++i) {

        for (int j = 0; j < BOARD\_SIZE; ++j) {

            if (board[i][j] == king) {

                return false;

            }

        }

    }

    return true;

}

int main() {

    initializeBoard();

    string move;

    bool whiteTurn = true;

    cout << "Welcome to Chess!" << endl;

    while (true) {

        displayBoard();

        if (checkWinCondition('k')) {

            cout << "White wins! Black's king is captured." << endl;

            break;

        }

        if (checkWinCondition('K')) {

            cout << "Black wins! White's king is captured." << endl;

            break;

        }

        cout << (whiteTurn ? "White's" : "Black's") << " turn. Enter your move (e.g., E2 E4): ";

        cin >> move;

        char fromCol = move[0];

        char fromRow = move[1];

        char toCol = move[3];

        char toRow = move[4];

        int fromRowIdx = 8 - (fromRow - '0');

        int fromColIdx = fromCol - 'A';

        int toRowIdx = 8 - (toRow - '0');

        int toColIdx = toCol - 'A';

        if (isValidPosition(fromRowIdx, fromColIdx) && isValidPosition(toRowIdx, toColIdx)) {

            board[toRowIdx][toColIdx] = board[fromRowIdx][fromColIdx];

            board[fromRowIdx][fromColIdx] = '.';

            whiteTurn = !whiteTurn;

        } else {

            cout << "Invalid move. Please try again." << endl;

        }

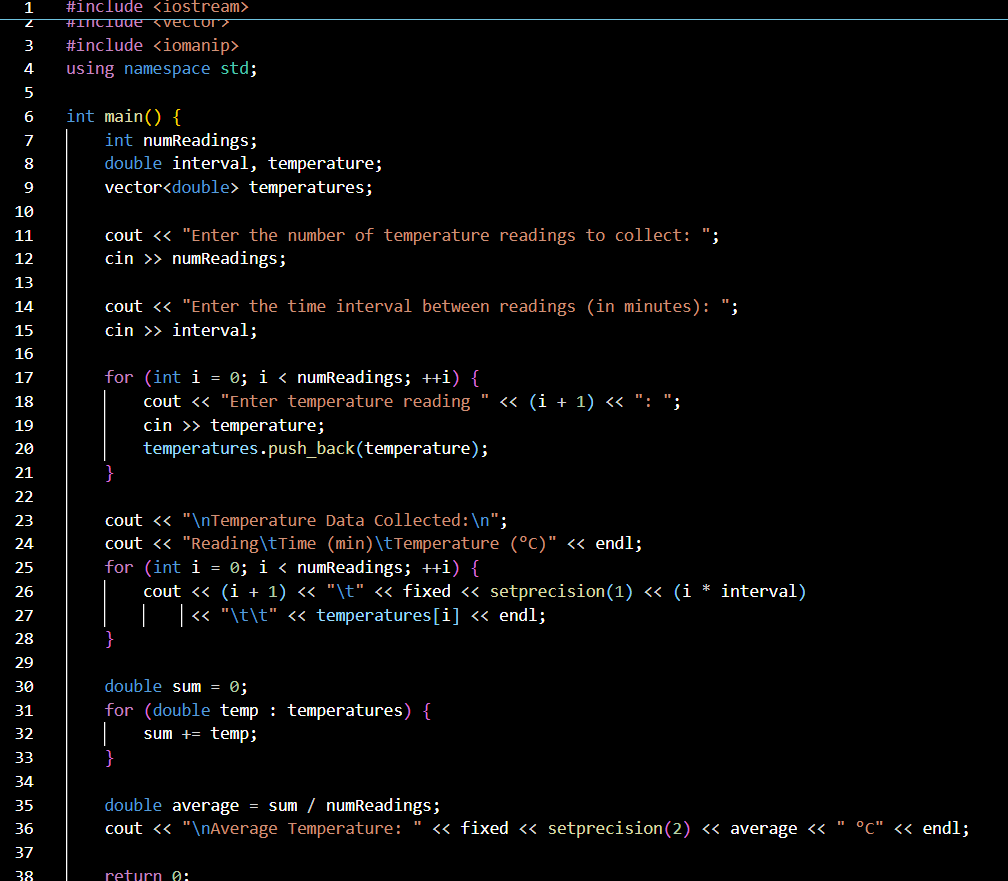
    }

    return 0;

}

**Week 5.**

**Case 01.**



**Case 02.**

#include <iostream>

#include <fstream>

#include <string>

#include <sstream>

#include <iomanip>

using namespace std;

struct SalesRecord {

    string product;

    int quantity;

    double price;

};

bool readData(const string& filename, SalesRecord records[], int& recordCount, int maxRecords) {

    ifstream inputFile(filename);

    if (!inputFile) {

        cerr << "Error: Unable to open file " << filename << endl;

        return false;

    }

    string line;

    recordCount = 0;

    while (getline(inputFile, line) && recordCount < maxRecords) {

        stringstream ss(line);

        getline(ss, records[recordCount].product, ',');

        ss >> records[recordCount].quantity;

        ss.ignore(1);

        ss >> records[recordCount].price;

        recordCount++;

    }

    inputFile.close();

    return true;

}

void writeResults(const string& filename, double totalSales) {

    ofstream outputFile(filename);

    if (!outputFile) {

        cerr << "Error: Unable to open file " << filename << endl;

        return;

    }

    outputFile << "Total Sales: $" << fixed << setprecision(2) << totalSales << endl;

    outputFile.close();

}

double calculateTotalSales(const SalesRecord records[], int recordCount) {

    double totalSales = 0;

    for (int i = 0; i < recordCount; ++i) {

        totalSales += records[i].quantity \* records[i].price;

    }

    return totalSales;

}

int main() {

    const int MAX\_RECORDS = 1000;

    SalesRecord records[MAX\_RECORDS];

    int recordCount = 0;

    string inputFilename, outputFilename;

    cout << "Enter the input file name: ";

    cin >> inputFilename;

    cout << "Enter the output file name: ";

    cin >> outputFilename;

    if (!readData(inputFilename, records, recordCount, MAX\_RECORDS)) {

        cout << "Failed to read data. Exiting program." << endl;

        return 1;

    }

    if (recordCount == 0) {

        cout << "No data to process. Exiting program." << endl;

        return 1;

    }

    double totalSales = calculateTotalSales(records, recordCount);

    cout << "Total Sales: $" << fixed << setprecision(2) << totalSales << endl;

    writeResults(outputFilename, totalSales);

    cout << "Results written to " << outputFilename << endl;

    return 0;

}

**Case 03.**

#include <iostream>

#include <stdexcept>

#include <string>

using namespace std;

double calculateTransaction(double amount, double rate) {

    if (rate == 0) {

        throw runtime\_error("Division by zero error: The rate cannot be zero.");

    }

    return amount / rate;

}

int main() {

    double amount, rate, result;

    string choice;

    cout << "Welcome to the Financial Transaction Processor" << endl;

    do {

        try {

            cout << "Enter the transaction amount: ";

            cin >> amount;

            if (cin.fail()) {

                throw runtime\_error("Invalid input: Please enter a numeric value for the amount.");

            }

            cout << "Enter the transaction rate: ";

            cin >> rate;

            if (cin.fail()) {

                throw runtime\_error("Invalid input: Please enter a numeric value for the rate.");

            }

            result = calculateTransaction(amount, rate);

            cout << "Transaction result: " << result << endl;

        } catch (const runtime\_error& e) {

            cout << "Error: " << e.what() << endl;

        }

        cout << "Do you want to process another transaction? (yes/no): ";

        cin >> choice;

    } while (choice == "yes");

    cout << "Thank you for using the Financial Transaction Processor. Goodbye!" << endl;

    return 0;

}

**Case 04.**

#include <iostream>

#include <fstream>

#include <string>

#include <filesystem>

using namespace std;

bool createFile(const string& filename) {

    ofstream file(filename);

    if (!file) {

        cout << "Error: Unable to create file " << filename << endl;

        return false;

    }

    cout << "File " << filename << " created successfully." << endl;

    return true;

}

bool writeFile(const string& filename, const string& content) {

    ofstream file(filename, ios::app); // Open file in append mode

    if (!file) {

        cout << "Error: Unable to write to file " << filename << endl;

        return false;

    }

    file << content << endl;

    cout << "Content written to " << filename << endl;

    return true;

}

bool deleteFile(const string& filename) {

    if (!filesystem::remove(filename)) {

        cout << "Error: Unable to delete file " << filename << endl;

        return false;

    }

    cout << "File " << filename << " deleted successfully." << endl;

    return true;

}

int main() {

    string filename, content, choice;

    cout << "Welcome to the File Management Application" << endl;

    do {

        cout << "\nMenu:\n";

        cout << "1. Create a file\n2. Write to a file\n3. Delete a file\n4. Exit\n";

        cout << "Enter your choice: ";

        cin >> choice;

        if (choice == "1") {

            cout << "Enter the file name to create: ";

            cin >> filename;

            createFile(filename);

        } else if (choice == "2") {

            cout << "Enter the file name to write to: ";

            cin >> filename;

            cout << "Enter the content to write: ";

            cin.ignore(); // Ignore leftover newline

            getline(cin, content);

            writeFile(filename, content);

        } else if (choice == "3") {

            cout << "Enter the file name to delete: ";

            cin >> filename;

            deleteFile(filename);

        } else if (choice == "4") {

            cout << "Exiting the application. Goodbye!" << endl;

            break;

        } else {

            cout << "Invalid choice. Please try again." << endl;

        }

    } while (true);

    return 0;

}

**Week 6.**

**Case 01.**

#include <iostream>

#include <string>

using namespace std;

void preparePasta() {

    cout << "Step 1: Boil water in a large pot." << endl;

    cout << "Step 2: Add pasta and cook for 10-12 minutes." << endl;

    cout << "Step 3: Drain the pasta and add sauce of your choice." << endl;

    cout << "Step 4: Serve with grated cheese and enjoy your pasta!" << endl;

}

void prepareSalad() {

    cout << "Step 1: Wash lettuce and chop into bite-sized pieces." << endl;

    cout << "Step 2: Add diced tomatoes, cucumbers, and olives." << endl;

    cout << "Step 3: Drizzle with dressing and toss well." << endl;

    cout << "Step 4: Serve fresh and enjoy your salad!" << endl;

}

void prepareBurger() {

    cout << "Step 1: Grill the burger patty until fully cooked." << endl;

    cout << "Step 2: Toast the burger buns." << endl;

    cout << "Step 3: Assemble the burger with lettuce, tomato, cheese, and condiments." << endl;

    cout << "Step 4: Serve hot and enjoy your burger!" << endl;

}

void prepareSoup() {

    cout << "Step 1: Chop vegetables and sauté in a pot with oil." << endl;

    cout << "Step 2: Add broth and bring to a boil." << endl;

    cout << "Step 3: Simmer for 20-30 minutes until the vegetables are tender." << endl;

    cout << "Step 4: Season with salt and pepper and serve hot." << endl;

}

int main() {

    string order;

    cout << "Welcome to the restaurant kitchen!" << endl;

    while (true) {

        cout << "Please enter the dish you want to prepare (Pasta, Salad, Burger, Soup) or 'exit' to quit: ";

        cin >> order;

        if (order == "Pasta") {

            preparePasta();

        } else if (order == "Salad") {

            prepareSalad();

        } else if (order == "Burger") {

            prepareBurger();

        } else if (order == "Soup") {

            prepareSoup();

        } else if (order == "exit") {

            cout << "Exiting the kitchen. Goodbye!" << endl;

            break;

        } else {

            cout << "Sorry, we don't serve that dish. Please choose from the available options." << endl;

        }

    }

    return 0;

}

**Week 7.**

**Case 01.**

#include <iostream>

using namespace std;

// Function for addition

double add(double a, double b) {

    return a + b;

}

// Function for subtraction

double subtract(double a, double b) {

    return a - b;

}

// Function for multiplication

double multiply(double a, double b) {

    return a \* b;

}

// Function for division

double divide(double a, double b) {

    if (b != 0) {

        return a / b;

    } else {

        cout << "Error: Division by zero!" << endl;

        return 0; // To avoid division by zero error

    }

}

int main() {

    double num1, num2;

    char op;

    cout << "Welcome to the Calculator Application!" << endl;

    cout << "Enter two numbers: ";

    cin >> num1 >> num2;

    cout << "Choose an operation (+, -, \*, /): ";

    cin >> op;

    double result = 0;

    // Switch to select the operation

    switch (op) {

        case '+':

            result = add(num1, num2);

            break;

        case '-':

            result = subtract(num1, num2);

            break;

        case '\*':

            result = multiply(num1, num2);

            break;

        case '/':

            result = divide(num1, num2);

            break;

        default:

            cout << "Invalid operation!" << endl;

            return 1; // Exit the program if the operation is invalid

    }

    cout << "Result: " << result << endl;

    return 0;

}

**Case 02.**

#include <iostream>

#include <string>

using namespace std;

struct Character {

    string name;

    int health;

    int attack;

    int defense;

};

void createCharacter(Character &charac) {

    cout << "Enter character name: ";

    cin >> charac.name;

    charac.health = 100;

    charac.attack = 20;

    charac.defense = 10;

    cout << "Character " << charac.name << " created with 100 health, 20 attack, and 10 defense!" << endl;

}

void printStatus(const Character &charac) {

    cout << "\nCharacter Status: " << endl;

    cout << "Name: " << charac.name << endl;

    cout << "Health: " << charac.health << endl;

    cout << "Attack: " << charac.attack << endl;

    cout << "Defense: " << charac.defense << endl;

}

void startQuest() {

    cout << "\nYou have accepted a quest to defeat the Dragon!" << endl;

    cout << "Prepare for battle!" << endl;

}

void battle(Character &player) {

    Character enemy;

    enemy.name = "Dragon";

    enemy.health = 150;

    enemy.attack = 30;

    enemy.defense = 15;

    cout << "\nA wild " << enemy.name << " appears!" << endl;

    while (player.health > 0 && enemy.health > 0) {

        cout << "\nYour turn to attack!" << endl;

        int damage = player.attack - enemy.defense;

        if (damage > 0) {

            enemy.health -= damage;

            cout << "You dealt " << damage << " damage to the " << enemy.name << endl;

        } else {

            cout << "Your attack was too weak to harm the " << enemy.name << endl;

        }

        if (enemy.health <= 0) {

            cout << "You have defeated the " << enemy.name << "!" << endl;

            break;

        }

        cout << "\nThe " << enemy.name << " attacks!" << endl;

        damage = enemy.attack - player.defense;

        if (damage > 0) {

            player.health -= damage;

            cout << "The " << enemy.name << " dealt " << damage << " damage to you!" << endl;

        } else {

            cout << "The " << enemy.name << "'s attack was blocked!" << endl;

        }

        if (player.health <= 0) {

            cout << "You have been defeated by the " << enemy.name << "!" << endl;

            break;

        }

        printStatus(player);

    }

}

int main() {

    Character player;

    int choice;

    cout << "Welcome to the Text-based RPG!" << endl;

    createCharacter(player);

    do {

        cout << "\nChoose an option:" << endl;

        cout << "1. View Character Status" << endl;

        cout << "2. Start Quest" << endl;

        cout << "3. Start Battle" << endl;

        cout << "4. Exit Game" << endl;

        cout << "Enter your choice: ";

        cin >> choice;

        switch (choice) {

            case 1:

                printStatus(player);

                break;

            case 2:

                startQuest();

                break;

            case 3:

                battle(player);

                break;

            case 4:

                cout << "Exiting game... Goodbye!" << endl;

                break;

            default:

                cout << "Invalid choice, try again!" << endl;

        }

    } while (choice != 4);

    return 0;

}

**Week 8.**

**Case 01.**

#include <iostream>

#include <limits>

using namespace std;

const int MAX\_STUDENTS = 30;  // Maximum number of students

void inputScores(int scores[], int &numStudents) {

    cout << "Enter the number of students: ";

    cin >> numStudents;

    if (numStudents > MAX\_STUDENTS) {

        cout << "Maximum number of students is " << MAX\_STUDENTS << endl;

        numStudents = MAX\_STUDENTS;

    }

    for (int i = 0; i < numStudents; ++i) {

        cout << "Enter score for student " << (i + 1) << ": ";

        cin >> scores[i];

    }

}

void printScores(int scores[], int numStudents) {

    cout << "\nScores of all students:" << endl;

    for (int i = 0; i < numStudents; ++i) {

        cout << "Student " << (i + 1) << ": " << scores[i] << endl;

    }

}

double calculateAverage(int scores[], int numStudents) {

    int total = 0;

    for (int i = 0; i < numStudents; ++i) {

        total += scores[i];

    }

    return (double)total / numStudents;

}

int findHighestScore(int scores[], int numStudents) {

    int highest = scores[0];

    for (int i = 1; i < numStudents; ++i) {

        if (scores[i] > highest) {

            highest = scores[i];

        }

    }

    return highest;

}

int findLowestScore(int scores[], int numStudents) {

    int lowest = scores[0];

    for (int i = 1; i < numStudents; ++i) {

        if (scores[i] < lowest) {

            lowest = scores[i];

        }

    }

    return lowest;

}

int main() {

    int scores[MAX\_STUDENTS];

    int numStudents;

    int choice;

    do {

        cout << "\nExam Score Management System" << endl;

        cout << "1. Input Scores" << endl;

        cout << "2. Display Scores" << endl;

        cout << "3. Calculate Average Score" << endl;

        cout << "4. Find Highest Score" << endl;

        cout << "5. Find Lowest Score" << endl;

        cout << "6. Exit" << endl;

        cout << "Enter your choice: ";

        cin >> choice;

        switch (choice) {

            case 1:

                inputScores(scores, numStudents);

                break;

            case 2:

                printScores(scores, numStudents);

                break;

            case 3: {

                double average = calculateAverage(scores, numStudents);

                cout << "Average Score: " << average << endl;

                break;

            }

            case 4: {

                int highest = findHighestScore(scores, numStudents);

                cout << "Highest Score: " << highest << endl;

                break;

            }

            case 5: {

                int lowest = findLowestScore(scores, numStudents);

                cout << "Lowest Score: " << lowest << endl;

                break;

            }

            case 6:

                cout << "Exiting the system. Goodbye!" << endl;

                break;

            default:

                cout << "Invalid choice, please try again." << endl;

        }

    } while (choice != 6);

    return 0;

}

**Case 02.**

#include <iostream>

#include <string>

using namespace std;

const int MAX\_PRODUCTS = 10;  // Maximum number of products in inventory

struct Product {

    string name;

    int quantity;

};

void addProduct(Product inventory[], int &numProducts) {

    if (numProducts >= MAX\_PRODUCTS) {

        cout << "Inventory is full! Cannot add more products." << endl;

        return;

    }

    cout << "Enter product name: ";

    cin >> inventory[numProducts].name;

    cout << "Enter quantity for " << inventory[numProducts].name << ": ";

    cin >> inventory[numProducts].quantity;

    numProducts++;

    cout << "Product added successfully!" << endl;

}

void updateStock(Product inventory[], int numProducts) {

    string productName;

    int newQuantity;

    cout << "Enter product name to update stock: ";

    cin >> productName;

    bool found = false;

    for (int i = 0; i < numProducts; ++i) {

        if (inventory[i].name == productName) {

            cout << "Enter new quantity for " << productName << ": ";

            cin >> newQuantity;

            inventory[i].quantity = newQuantity;

            found = true;

            cout << "Stock updated for " << productName << "." << endl;

            break;

        }

    }

    if (!found) {

        cout << "Product not found in inventory." << endl;

    }

}

void removeProduct(Product inventory[], int &numProducts) {

    string productName;

    cout << "Enter product name to remove: ";

    cin >> productName;

    bool found = false;

    for (int i = 0; i < numProducts; ++i) {

        if (inventory[i].name == productName) {

            // Shift the remaining products to remove the product

            for (int j = i; j < numProducts - 1; ++j) {

                inventory[j] = inventory[j + 1];

            }

            numProducts--;

            found = true;

            cout << productName << " removed from inventory." << endl;

            break;

        }

    }

    if (!found) {

        cout << "Product not found in inventory." << endl;

    }

}

void viewInventory(Product inventory[], int numProducts) {

    if (numProducts == 0) {

        cout << "Inventory is empty!" << endl;

        return;

    }

    cout << "\nCurrent Inventory:\n";

    for (int i = 0; i < numProducts; ++i) {

        cout << "Product Name: " << inventory[i].name << ", Quantity: " << inventory[i].quantity << endl;

    }

}

int main() {

    Product inventory[MAX\_PRODUCTS];

    int numProducts = 0;  // Number of products in the inventory

    int choice;

    do {

        cout << "\nInventory Management System" << endl;

        cout << "1. Add Product" << endl;

        cout << "2. Update Stock" << endl;

        cout << "3. Remove Product" << endl;

        cout << "4. View Inventory" << endl;

        cout << "5. Exit" << endl;

        cout << "Enter your choice: ";

        cin >> choice;

        switch (choice) {

            case 1:

                addProduct(inventory, numProducts);

                break;

            case 2:

                updateStock(inventory, numProducts);

                break;

            case 3:

                removeProduct(inventory, numProducts);

                break;

            case 4:

                viewInventory(inventory, numProducts);

                break;

            case 5:

                cout << "Exiting the system. Goodbye!" << endl;

                break;

            default:

                cout << "Invalid choice! Please try again." << endl;

        }

    } while (choice != 5);

    return 0;

}

**Case 03.**

#include <iostream>

#include <string>

using namespace std;

const int MAX\_RESPONSES = 100;  // Maximum number of survey responses

void getSurveyResponses(string responses[], int &numResponses) {

    cout << "Enter the number of survey responses (max " << MAX\_RESPONSES << "): ";

    cin >> numResponses;

    if (numResponses < 1 || numResponses > MAX\_RESPONSES) {

        cout << "Invalid number of responses. Please enter a number between 1 and " << MAX\_RESPONSES << "." << endl;

        numResponses = 0;  // Reset numResponses in case of invalid input

        return;

    }

    // Clear input buffer before reading strings

    cin.ignore();

    for (int i = 0; i < numResponses; ++i) {

        cout << "Enter response " << (i + 1) << ": ";

        getline(cin, responses[i]);

    }

}

void displaySurveyResponses(const string responses[], int numResponses) {

    cout << "\nSurvey Responses:" << endl;

    for (int i = 0; i < numResponses; ++i) {

        cout << "Response " << (i + 1) << ": " << responses[i] << endl;

    }

}

int main() {

    string surveyResponses[MAX\_RESPONSES];

    int numResponses;

    // Get survey responses safely

    getSurveyResponses(surveyResponses, numResponses);

    // If valid responses were entered, display them

    if (numResponses > 0) {

        displaySurveyResponses(surveyResponses, numResponses);

    }

    return 0;

}

**Week 9.**

**Case 01.**

#include <iostream>

#include <string>

using namespace std;

const int MAX\_STUDENTS = 5;

double calculateAverage(int grades[], int numStudents) {

    int sum = 0;

    for (int i = 0; i < numStudents; ++i) {

        sum += grades[i];

    }

    return static\_cast<double>(sum) / numStudents;

}

void displayStudentInfo(string names[], int ids[], int grades[], int numStudents) {

    cout << "\nStudent Records:\n";

    for (int i = 0; i < numStudents; ++i) {

        cout << "Student Name: " << names[i] << ", ID: " << ids[i] << ", Grade: " << grades[i] << endl;

    }

}

int main() {

    string studentNames[MAX\_STUDENTS];

    int studentIDs[MAX\_STUDENTS];

    int studentGrades[MAX\_STUDENTS];

    int numStudents;

    cout << "Enter the number of students (max " << MAX\_STUDENTS << "): ";

    cin >> numStudents;

    if (numStudents < 1 || numStudents > MAX\_STUDENTS) {

        cout << "Invalid number of students. Please enter a number between 1 and " << MAX\_STUDENTS << "." << endl;

        return 1;

    }

    for (int i = 0; i < numStudents; ++i) {

        cout << "Enter name for student " << (i + 1) << ": ";

        cin.ignore();

        getline(cin, studentNames[i]);

        cout << "Enter ID for student " << (i + 1) << ": ";

        cin >> studentIDs[i];

        cout << "Enter grade for student " << (i + 1) << ": ";

        cin >> studentGrades[i];

    }

    displayStudentInfo(studentNames, studentIDs, studentGrades, numStudents);

    double average = calculateAverage(studentGrades, numStudents);

    cout << "\nAverage grade for the class: " << average << endl;

    return 0;

}

**Case 02.**

#include <iostream>

#include <string>

using namespace std;

const int MAX\_WEEKS = 5;  // Maximum number of weeks in a month

const int DAYS\_IN\_WEEK = 7; // Number of days in a week

void displayCalendar(string calendar[MAX\_WEEKS][DAYS\_IN\_WEEK], int daysInMonth) {

    cout << "\nCalendar for the Month:\n";

    int day = 1;

    for (int i = 0; i < MAX\_WEEKS; ++i) {

        for (int j = 0; j < DAYS\_IN\_WEEK; ++j) {

            if (day <= daysInMonth) {

                cout << day << ": " << calendar[i][j] << "\t";

                ++day;

            } else {

                cout << "\t";

            }

        }

        cout << endl;

    }

}

void addEvent(string calendar[MAX\_WEEKS][DAYS\_IN\_WEEK], int day, string event) {

    int week = (day - 1) / DAYS\_IN\_WEEK;

    int dayOfWeek = (day - 1) % DAYS\_IN\_WEEK;

    calendar[week][dayOfWeek] = event;

}

int main() {

    string calendar[MAX\_WEEKS][DAYS\_IN\_WEEK] = {};

    int daysInMonth, day;

    string event;

    cout << "Enter the number of days in the month (1-31): ";

    cin >> daysInMonth;

    if (daysInMonth < 1 || daysInMonth > 31) {

        cout << "Invalid number of days." << endl;

        return 1;

    }

    do {

        displayCalendar(calendar, daysInMonth);

        cout << "\nEnter a day (1 to " << daysInMonth << ") to add an event (0 to stop): ";

        cin >> day;

        if (day == 0) break;

        if (day < 1 || day > daysInMonth) {

            cout << "Invalid day." << endl;

            continue;

        }

        cin.ignore(); // To ignore the newline character left by cin

        cout << "Enter event description for day " << day << ": ";

        getline(cin, event);

        addEvent(calendar, day, event);

    } while (true);

    return 0;

}

**Case 03.**

#include <iostream>

#include <string>

using namespace std;

const int MAX\_EMPLOYEES = 5;

void displayEmployeeInfo(string names[], int ids[], double salaries[], string departments[], int numEmployees) {

    cout << "\nEmployee Information:\n";

    for (int i = 0; i < numEmployees; ++i) {

        cout << "Employee " << (i + 1) << ":\n";

        cout << "Name: " << names[i] << endl;

        cout << "ID: " << ids[i] << endl;

        cout << "Salary: $" << salaries[i] << endl;

        cout << "Department: " << departments[i] << endl;

        cout << "-----------------------------\n";

    }

}

void addEmployee(string names[], int ids[], double salaries[], string departments[], int &numEmployees) {

    if (numEmployees >= MAX\_EMPLOYEES) {

        cout << "Cannot add more employees, maximum limit reached." << endl;

        return;

    }

    cout << "Enter name of employee: ";

    cin.ignore();

    getline(cin, names[numEmployees]);

    cout << "Enter ID of employee: ";

    cin >> ids[numEmployees];

    cout << "Enter salary of employee: ";

    cin >> salaries[numEmployees];

    cout << "Enter department of employee: ";

    cin.ignore();

    getline(cin, departments[numEmployees]);

    numEmployees++;

}

int main() {

    string employeeNames[MAX\_EMPLOYEES];

    int employeeIDs[MAX\_EMPLOYEES];

    double employeeSalaries[MAX\_EMPLOYEES];

    string employeeDepartments[MAX\_EMPLOYEES];

    int numEmployees = 0;

    int choice;

    do {

        cout << "Employee Management System\n";

        cout << "1. Add Employee\n";

        cout << "2. Display All Employees\n";

        cout << "3. Exit\n";

        cout << "Enter your choice: ";

        cin >> choice;

        switch (choice) {

            case 1:

                addEmployee(employeeNames, employeeIDs, employeeSalaries, employeeDepartments, numEmployees);

                break;

            case 2:

                displayEmployeeInfo(employeeNames, employeeIDs, employeeSalaries, employeeDepartments, numEmployees);

                break;

            case 3:

                cout << "Exiting the Employee Management System.\n";

                break;

            default:

                cout << "Invalid choice. Please try again.\n";

        }

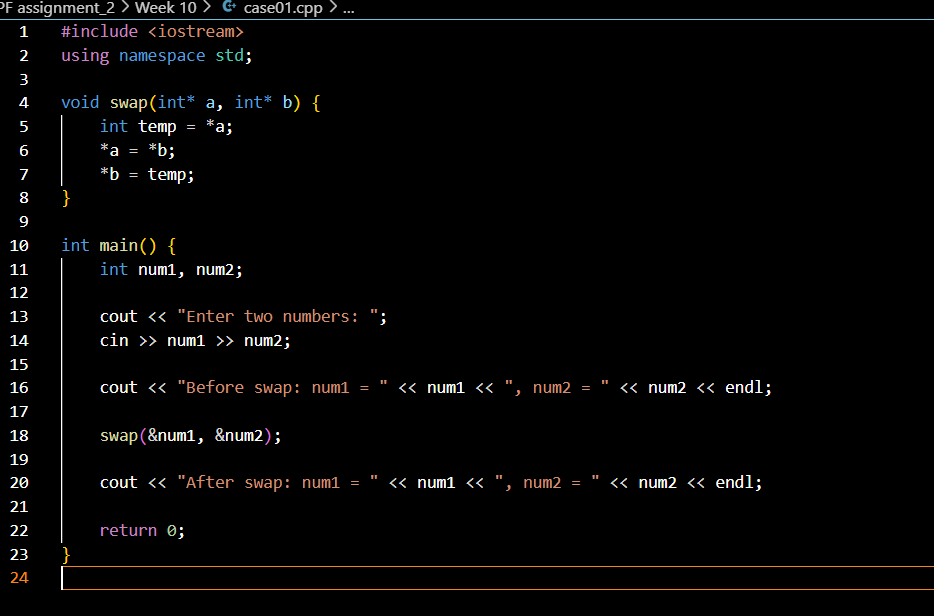
    } while (choice != 3);

    return 0;

}

**Week 10.**

**Case 01.**



**Case 02.**

#include <iostream>

#include <cstring>

using namespace std;

void createDocument(char\*& text, int size) {

    // Allocate memory dynamically for the document text

    text = new char[size + 1];  // +1 for null-terminator

}

void editDocument(char\* text, const char\* newText) {

    // Copy new text into the allocated memory

    strcpy(text, newText);

}

void displayDocument(const char\* text) {

    // Display the document content

    cout << "Document Content: " << text << endl;

}

void deleteDocument(char\*& text) {

    // Free the dynamically allocated memory

    delete[] text;

    text = nullptr;

}

int main() {

    char\* document = nullptr;

    int size;

    cout << "Enter the size of the document: ";

    cin >> size;

    createDocument(document, size);

    cout << "Enter the document text: ";

    cin.ignore();  // To clear the buffer before reading a line

    cin.getline(document, size + 1);  // Reading the input into the allocated memory

    displayDocument(document);

    cout << "Enter new text to edit the document: ";

    cin.getline(document, size + 1);  // Edit the document

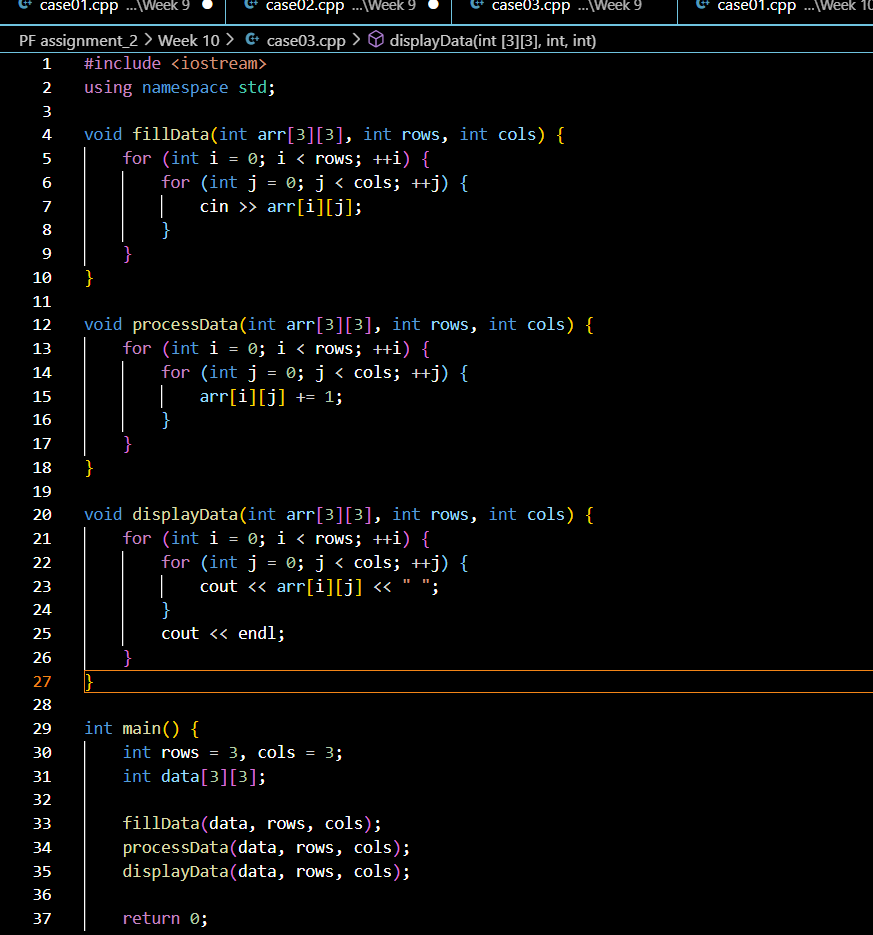
    displayDocument(document);

    deleteDocument(document);

    return 0;

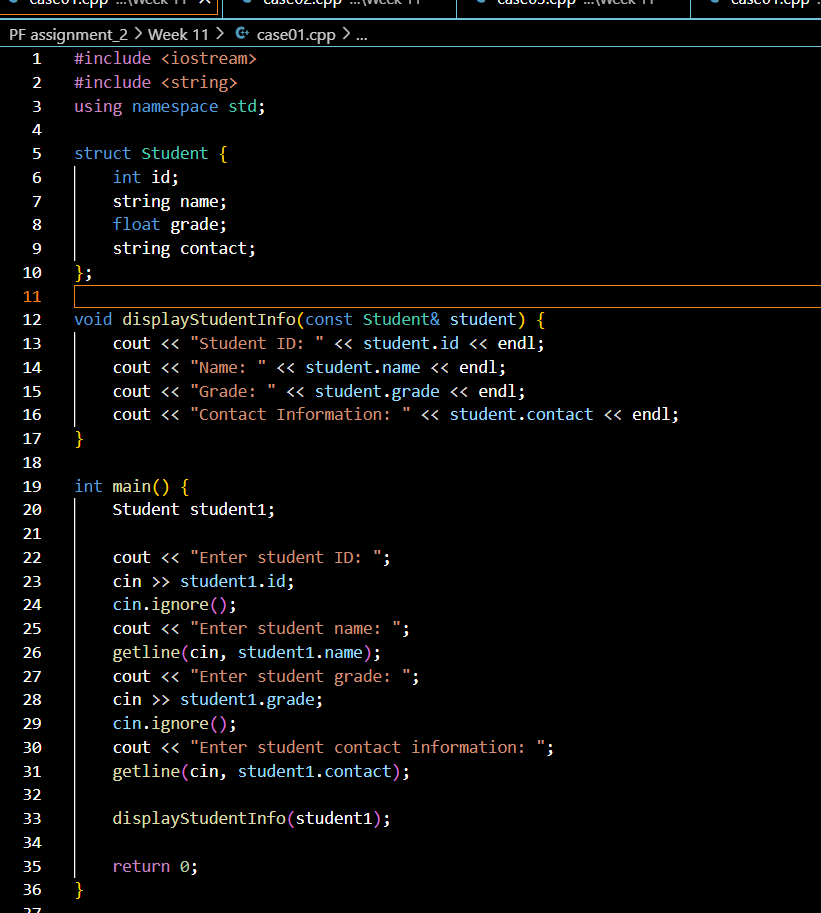
}

**Case 03.**

****

**Week 11.**

**Case 01.**

****

**Case 02.**

#include <iostream>

#include <string>

using namespace std;

struct Product {

    int id;

    string name;

    float price;

    int stockLevel;

};

void displayProductInfo(const Product& product) {

    cout << "Product ID: " << product.id << endl;

    cout << "Product Name: " << product.name << endl;

    cout << "Price: $" << product.price << endl;

    cout << "Stock Level: " << product.stockLevel << endl;

}

int main() {

    const int numProducts = 3;

    Product products[numProducts];

    for (int i = 0; i < numProducts; ++i) {

        cout << "Enter product " << i + 1 << " details:" << endl;

        cout << "Enter product ID: ";

        cin >> products[i].id;

        cin.ignore();

        cout << "Enter product name: ";

        getline(cin, products[i].name);

        cout << "Enter product price: ";

        cin >> products[i].price;

        cout << "Enter stock level: ";

        cin >> products[i].stockLevel;

        cout << endl;

    }

    cout << "Product Information:" << endl;

    for (int i = 0; i < numProducts; ++i) {

        displayProductInfo(products[i]);

        cout << endl;

    }

    return 0;

}

**Week 12.**

**Case 01.**

#include <iostream>

#include <string>

using namespace std;

struct Employee {

    int id;

    string name;

    float salary;

    string department;

};

void addEmployee(Employee& emp) {

    cout << "Enter employee ID: ";

    cin >> emp.id;

    cin.ignore();

    cout << "Enter employee name: ";

    getline(cin, emp.name);

    cout << "Enter employee salary: ";

    cin >> emp.salary;

    cin.ignore();

    cout << "Enter employee department: ";

    getline(cin, emp.department);

}

void updateSalary(Employee& emp, float newSalary) {

    emp.salary = newSalary;

    cout << "Salary updated to: " << emp.salary << endl;

}

void updateDepartment(Employee& emp, const string& newDepartment) {

    emp.department = newDepartment;

    cout << "Department updated to: " << emp.department << endl;

}

void displayEmployeeInfo(const Employee& emp) {

    cout << "Employee ID: " << emp.id << endl;

    cout << "Employee Name: " << emp.name << endl;

    cout << "Employee Salary: " << emp.salary << endl;

    cout << "Employee Department: " << emp.department << endl;

}

int main() {

    Employee employees[3];

    int choice;

    int empIndex;

    while (true) {

        cout << "\nEmployee Management System" << endl;

        cout << "1. Add New Employee" << endl;

        cout << "2. Update Employee Salary" << endl;

        cout << "3. Update Employee Department" << endl;

        cout << "4. Display Employee Info" << endl;

        cout << "5. Exit" << endl;

        cout << "Enter your choice: ";

        cin >> choice;

        switch (choice) {

            case 1:

                cout << "Enter employee index (0-2): ";

                cin >> empIndex;

                if (empIndex >= 0 && empIndex < 3) {

                    addEmployee(employees[empIndex]);

                } else {

                    cout << "Invalid index!" << endl;

                }

                break;

            case 2:

                cout << "Enter employee index (0-2): ";

                cin >> empIndex;

                if (empIndex >= 0 && empIndex < 3) {

                    float newSalary;

                    cout << "Enter new salary: ";

                    cin >> newSalary;

                    updateSalary(employees[empIndex], newSalary);

                } else {

                    cout << "Invalid index!" << endl;

                }

                break;

            case 3:

                cout << "Enter employee index (0-2): ";

                cin >> empIndex;

                if (empIndex >= 0 && empIndex < 3) {

                    string newDepartment;

                    cout << "Enter new department: ";

                    cin.ignore();

                    getline(cin, newDepartment);

                    updateDepartment(employees[empIndex], newDepartment);

                } else {

                    cout << "Invalid index!" << endl;

                }

                break;

            case 4:

                cout << "Enter employee index (0-2): ";

                cin >> empIndex;

                if (empIndex >= 0 && empIndex < 3) {

                    displayEmployeeInfo(employees[empIndex]);

                } else {

                    cout << "Invalid index!" << endl;

                }

                break;

            case 5:

                cout << "Exiting program." << endl;

                return 0;

            default:

                cout << "Invalid choice!" << endl;

        }

    }

    return 0;

}

**Case 02.**



**Week 13.**

**Case 01.**

#include <iostream>

#include <cctype>

#include <string>

using namespace std;

void convertToUpper(string& str) {

    for (char& c : str) {

        c = toupper(c);

    }

}

void convertToLower(string& str) {

    for (char& c : str) {

        c = tolower(c);

    }

}

void displayString(const string& str) {

    cout << "Processed String: " << str << endl;

}

int main() {

    string inputStr;

    int choice;

    cout << "Enter a string: ";

    getline(cin, inputStr);

    cout << "Choose an operation:\n";

    cout << "1. Convert to Uppercase\n";

    cout << "2. Convert to Lowercase\n";

    cout << "Enter your choice: ";

    cin >> choice;

    switch (choice) {

        case 1:

            convertToUpper(inputStr);

            break;

        case 2:

            convertToLower(inputStr);

            break;

        default:

            cout << "Invalid choice!" << endl;

            return 1;

    }

    displayString(inputStr);

    return 0;

}

**Week 14.**

**Case 01.**

#include <iostream>

#include <string>

using namespace std;

struct Contact {

    string name;

    string phoneNumber;

};

void addContact(Contact contacts[], int& count) {

    cout << "Enter contact name: ";

    cin.ignore();

    getline(cin, contacts[count].name);

    cout << "Enter contact phone number: ";

    getline(cin, contacts[count].phoneNumber);

    count++;

}

void displayContacts(const Contact contacts[], int count) {

    if (count == 0) {

        cout << "No contacts available." << endl;

        return;

    }

    for (int i = 0; i < count; i++) {

        cout << "Name: " << contacts[i].name << ", Phone Number: " << contacts[i].phoneNumber << endl;

    }

}

void searchContact(const Contact contacts[], int count, const string& searchName) {

    bool found = false;

    for (int i = 0; i < count; i++) {

        if (contacts[i].name == searchName) {

            cout << "Found contact: Name: " << contacts[i].name << ", Phone Number: " << contacts[i].phoneNumber << endl;

            found = true;

            break;

        }

    }

    if (!found) {

        cout << "Contact not found." << endl;

    }

}

void sortContacts(Contact contacts[], int count) {

    for (int i = 0; i < count - 1; i++) {

        for (int j = i + 1; j < count; j++) {

            if (contacts[i].name > contacts[j].name) {

                swap(contacts[i], contacts[j]);

            }

        }

    }

}

int main() {

    Contact contacts[100];

    int count = 0;

    int choice;

    while (true) {

        cout << "\nPhonebook Application\n";

        cout << "1. Add Contact\n";

        cout << "2. Display Contacts\n";

        cout << "3. Search Contact\n";

        cout << "4. Sort Contacts\n";

        cout << "5. Exit\n";

        cout << "Enter your choice: ";

        cin >> choice;

        switch (choice) {

            case 1:

                addContact(contacts, count);

                break;

            case 2:

                displayContacts(contacts, count);

                break;

            case 3: {

                string searchName;

                cout << "Enter name to search: ";

                cin.ignore();

                getline(cin, searchName);

                searchContact(contacts, count, searchName);

                break;

            }

            case 4:

                sortContacts(contacts, count);

                cout << "Contacts sorted." << endl;

                break;

            case 5:

                cout << "Exiting phonebook." << endl;

                return 0;

            default:

                cout << "Invalid choice!" << endl;

        }

    }

    return 0;

}

**Case 02.**

#include <iostream>

#include <string>

using namespace std;

struct Student {

    int id;

    string name;

    float grade;

};

void searchStudentById(const Student students[], int size, int searchId) {

    bool found = false;

    for (int i = 0; i < size; i++) {

        if (students[i].id == searchId) {

            cout << "Student found: ID: " << students[i].id << ", Name: " << students[i].name << ", Grade: " << students[i].grade << endl;

            found = true;

            break;

        }

    }

    if (!found) {

        cout << "Student with ID " << searchId << " not found." << endl;

    }

}

int main() {

    Student students[] = {

        {101, "Alice", 90.5},

        {102, "Bob", 85.3},

        {103, "Charlie", 88.7},

        {104, "David", 92.0}

    };

    int size = sizeof(students) / sizeof(students[0]);

    int searchId;

    cout << "Enter student ID to search: ";

    cin >> searchId;

    searchStudentById(students, size, searchId);

    return 0;

}