National University of Computer and Emerging Science Islamabad

Bilal Rasheed

Task 01:

```
#include <iostream>
#include <cstdlib>
#include <ctime>     // For time() function
using namespace std;
int main()
{

    system("color 2E");
    srand(time(0));

    // Generate a random number between 1 and 100
    int secretNumber = rand() % 100 + 1;

    int guess, attempts = 0;

    cout << "\tWelcome to the Guess the Number game!\n";
    cout << "\tTry to guess the number between 1 and 100.\n";

    do
    {
        cout << "\tEnter your guess: ";
        cin >> guess;

        attempts++;

        if (guess == secretNumber)
        {
            cout << "\n\a\tCongratulations! You guessed the
correct number in " << attempts << " attempts.\n";
        }
        else if (guess < secretNumber)
        {
            cout << "\tToo low. Try again.\n";
        }
        else
        {
            cout << "\tToo high. Try again.\n";
        }

    }
    while (guess != secretNumber);

    system("pause");
    return 0;
}
```

Task 02:

```
#include <iostream>

using namespace std;
int main()
{
    system("color 6F");
    double num1, num2;
    char operation;

    cout << "Enter the first number: ";
    cin >> num1;

    cout << "Enter the second number: ";
    cin >> num2;

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

    switch (operation)
    {
    case '+':
        cout << num1 << " + " << num2 << " = " << num1 + num2 <<
"\n";
        break;
    case '-':
        cout << num1 << " - " << num2 << " = " << num1 - num2 <<
"\n";
        break;
    case '*':
        cout << num1 << " * " << num2 << " = " << num1 * num2 <<
"\n";
        break;
    case '/':
        if (num2 != 0)
        {
            cout << num1 << " / " << num2 << " = " << num1 / num2
<< "\n";
        }
        else
        {
            cout << "Error: Division by zero is undefined.\n";
        }
        break;
    default:
        cout << "Error: Invalid operation.\n";
    }
    system("pause");
    return 0;
}
```

Task 03:

```
#include <iostream>
#include<cmath>
void displayBoard(char board[3][3]);
bool makeMove(char board[3][3], char currentPlayer);
bool checkWin(char board[3][3], char currentPlayer);
bool checkDraw(char board[3][3]);
void switchPlayers(char& currentPlayer);
bool playAgain();

using namespace std;
int main()
{
    char board[3][3] = { { ' ', ' ', ' ' }, { ' ', ' ', ' ' }, { '
', ' ', ' ' } };
    char currentPlayer = 'X';

    cout << "Welcome to Tic-Tac-Toe game!\n";

    do
    {

        displayBoard(board);
        while (!makeMove(board, currentPlayer));
        if (checkWin(board, currentPlayer))
        {
            displayBoard(board);
            cout << "Player " << currentPlayer << " wins!\n";
            if (!playAgain())
```

```cpp
            {
                break;
            }
            else
            {
                for (int i = 0; i < 10; ++i)
                {
                    for (int j = 0; j <10; ++j)
                    {
                        board[i][j] = ' ';
                    }
                }
                switchPlayers(currentPlayer);
            }
        }
        if (checkDraw(board))
        {
            displayBoard(board);
            cout << "It's a draw!\n";
            if (!playAgain())
            {
                break;
            }
            else
            {
                for (int i = 0; i < 3; ++i)
                {
                    for (int j = 0; j < 3; ++j)
                    {
                        board[i][j] = ' ';
                    }
                }
                switchPlayers(currentPlayer);
            }
        }
        switchPlayers(currentPlayer);

    }
    while (true);

    return 0;
}

void displayBoard(char board[3][3])
{
    cout << "\n  1 2 3\n";
    for (int i = 0; i < 3; ++i)
    {
        cout << i + 1 << " ";
        for (int j = 0; j < 3; ++j)
        {
            cout << board[i][j] << " ";
        }
        cout << "\n";
    }
}

bool makeMove(char board[3][3], char currentPlayer)
{
    int row, col;
    cout << "Player " << currentPlayer << ", enter your move (row
and column): ";
    cin >> row >> col;

    // Adjust row and column to 0-based index
    --row;
    --col;

    // Check if the chosen cell is valid and not already taken
    if (row >= 0 && row < 3 && col >= 0 && col < 3 &&
board[row][col] == ' ')
    {
        board[row][col] = currentPlayer;
        return true;
    }
    else
    {
        cout << "Invalid move. Try again.\n";
        return false;
    }
}

bool checkWin(char board[3][3], char currentPlayer)
{
    // Check rows, columns, and diagonals for a win
    for (int i = 0; i < 3; ++i)
    {
        if ((board[i][0] == currentPlayer && board[i][1] ==
currentPlayer && board[i][2] == currentPlayer) ||
            (board[0][i] == currentPlayer && board[1][i] ==
currentPlayer && board[2][i] == currentPlayer))
        {
            return true;
        }
    }

    if ((board[0][0] == currentPlayer && board[1][1] ==
currentPlayer && board[2][2] == currentPlayer) ||
        (board[0][2] == currentPlayer && board[1][1] ==
currentPlayer && board[2][0] == currentPlayer))
    {
        return true;
    }

    return false;
}

bool checkDraw(char board[3][3])
{
    // Check if the board is full
    for (int i = 0; i < 3; ++i)
    {
        for (int j = 0; j < 3; ++j)
        {
            if (board[i][j] == ' ')
            {
                return false; // If there is an empty space, the
game is not a draw
            }
        }
    }

    return true; // The board is full, and no player has won, so
it's a draw
}

void switchPlayers(char& currentPlayer)
{
    currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';
}

bool playAgain()
{
    char choice;
    cout << "Do you want to play again? (y/n): ";
    cin >> choice;
    return (choice == 'y' || choice == 'Y');
}
```

Task 04:

```
#include <iostream>
#include <string>

using namespace std;

const int MAX_TASKS = 100;

class TaskManager
{
public:
    virtual ~TaskManager() {}
    virtual void display() const = 0;
    virtual bool isCompleted() const = 0;
    virtual void markAsCompleted() = 0;
};
class RegularTask : public TaskManager
{
private:
    string description;

public:
    RegularTask(const string& desc) : description(desc) {}

    void display() const override
    {
        cout << description << " - Pending\n";
    }

    bool isCompleted() const override
    {
        return false;
    }

    void markAsCompleted() override
    {

        cout << "Cannot mark a regular task as completed.\n";
    }
};
class CompletedTask : public TaskManager
{
private:
```

```cpp
    string description;

public:
    CompletedTask(const string& desc) : description(desc) {}

    void display() const override
    {
        cout << description << " - Completed\n";
    }

    bool isCompleted() const override
    {
        return true;
    }
    void markAsCompleted() override
    {
        cout << "Task is already marked as completed.\n";
    }
};

// Function prototypes
void displayMenu();
void addTask(TaskManager* tasks[], int& taskCount);
void viewTasks(const TaskManager* tasks[], int taskCount);
void markTaskAsCompleted(TaskManager* tasks[], int taskCount);
void removeTask(TaskManager* tasks[], int& taskCount);

int main()
{
    TaskManager* tasks[MAX_TASKS];
    int taskCount = 0;

    while (true)
    {
        displayMenu();

        int choice;
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
        case 1:
            addTask(tasks, taskCount);
            break;
        case 2:
            viewTasks(tasks, taskCount);
```

```cpp
                break;
            case 3:
                markTaskAsCompleted(tasks, taskCount);
                break;
            case 4:
                removeTask(tasks, taskCount);
                break;
            case 5:
                cout << "Exiting the to-do list manager. Goodbye!\n";

                for (int i = 0; i < taskCount; ++i)
                {
                    delete tasks[i];
                }

                return 0;
            default:
                cout << "Invalid choice. Please try again.\n";
            }
        }

    return 0;
}

void displayMenu()
{
    cout << "\n===== To-Do List Manager =====\n";
    cout << "1. Add Task\n";
    cout << "2. View Tasks\n";
    cout << "3. Mark Task as Completed\n";
    cout << "4. Remove Task\n";
    cout << "5. Exit\n";
}

void addTask(TaskManager* tasks[], int& taskCount)
{
    if (taskCount < MAX_TASKS) {
        string description;
        cout << "Enter task description: ";

          cin.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');

        getline(cin, description);
        tasks[taskCount] = new RegularTask(description);
        taskCount++;
        cout << "Task added successfully!\n";
    }
    else
    {
        cout << "Task limit reached. Unable to add more tasks.\n";
    }
}

void viewTasks(const TaskManager* tasks[], int taskCount)
{
    if (taskCount == 0)
    {
        cout << "No tasks available. Add tasks to view.\n";
    }
    else
    {
        cout << "\n===== Tasks =====\n";
        for (int i = 0; i < taskCount; ++i)
        {
            tasks[i]->display();
        }
    }
}

void markTaskAsCompleted(TaskManager* tasks[], int taskCount)
{
    viewTasks(tasks, taskCount);

    if (taskCount == 0)
    {
        return;
    }

    int taskNumber;
    cout << "Enter the number of the task to mark as completed: ";
    cin >> taskNumber;

    if (taskNumber >= 1 && taskNumber <= taskCount)
    {
        RegularTask* regularTask =
dynamic_cast<RegularTask*>(tasks[taskNumber - 1]);
        if (regularTask)
        {
            delete tasks[taskNumber - 1];
            tasks[taskNumber - 1] = new CompletedTask(regularTask-
>description);
```

```
            cout << "Task marked as completed!\n";
        }
        else
        {
            cout << "Cannot mark a completed task as completed 
again.\n";
        }
    }
    else
    {
        cout << "Invalid task number. No task marked as 
completed.\n";
    }
}

void removeTask(TaskManager* tasks[], int& taskCount) {
    viewTasks(tasks, taskCount);

    if (taskCount == 0) {
        return;
    }

    int taskNumber;
    cout << "Enter the number of the task to remove: ";
    cin >> taskNumber;

    if (taskNumber >= 1 && taskNumber <= taskCount) {

        delete tasks[taskNumber - 1];
        for (int i = taskNumber - 1; i < taskCount - 1; ++i)
        {
            tasks[i] = tasks[i + 1];
        }

        taskCount--;
        cout << "Task removed successfully!\n";
    }
    else
    {
        cout << "Invalid task number. No task removed.\n";
    }
}
```