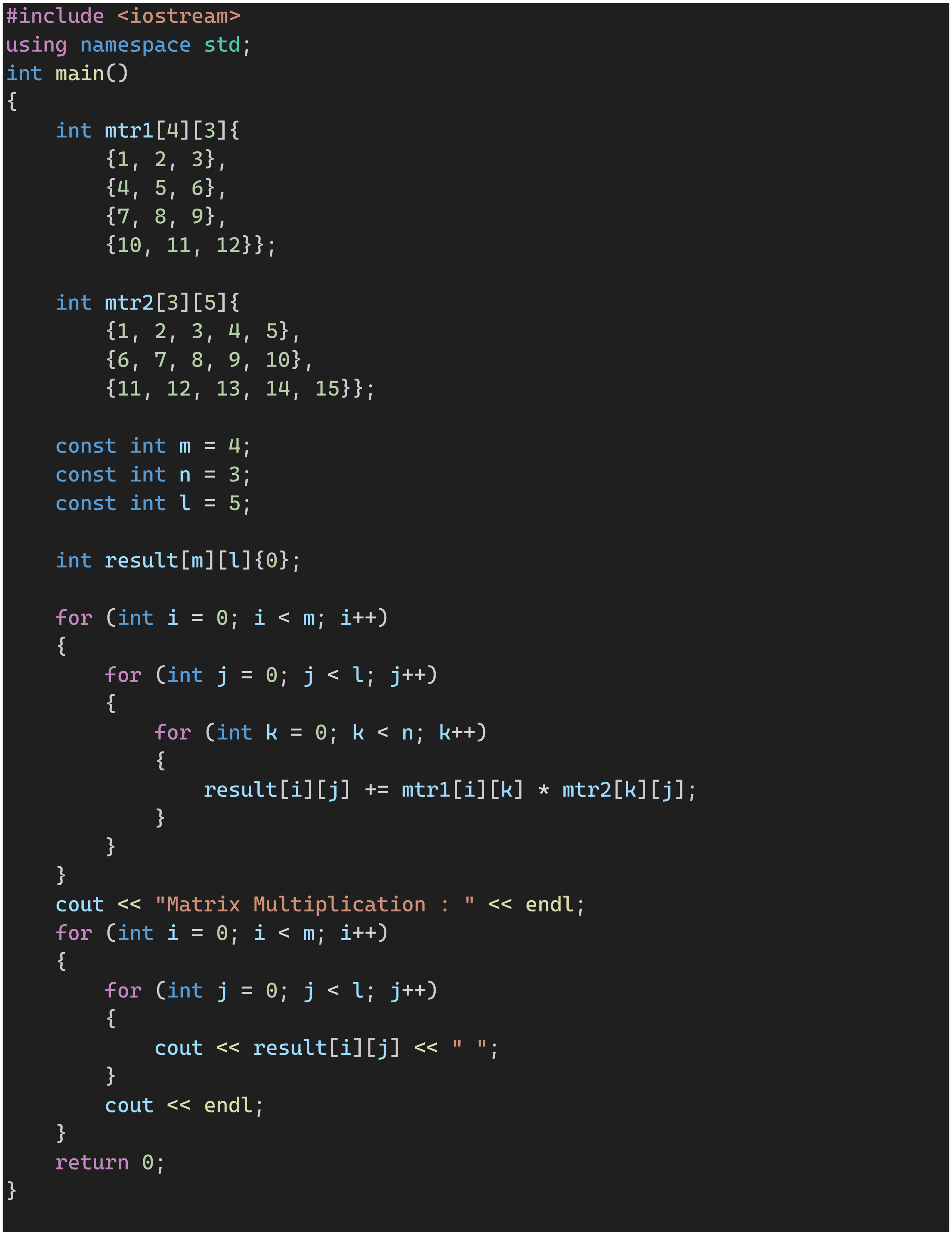
C

Assignment – 3

Name – Kishan R Vaghamashi

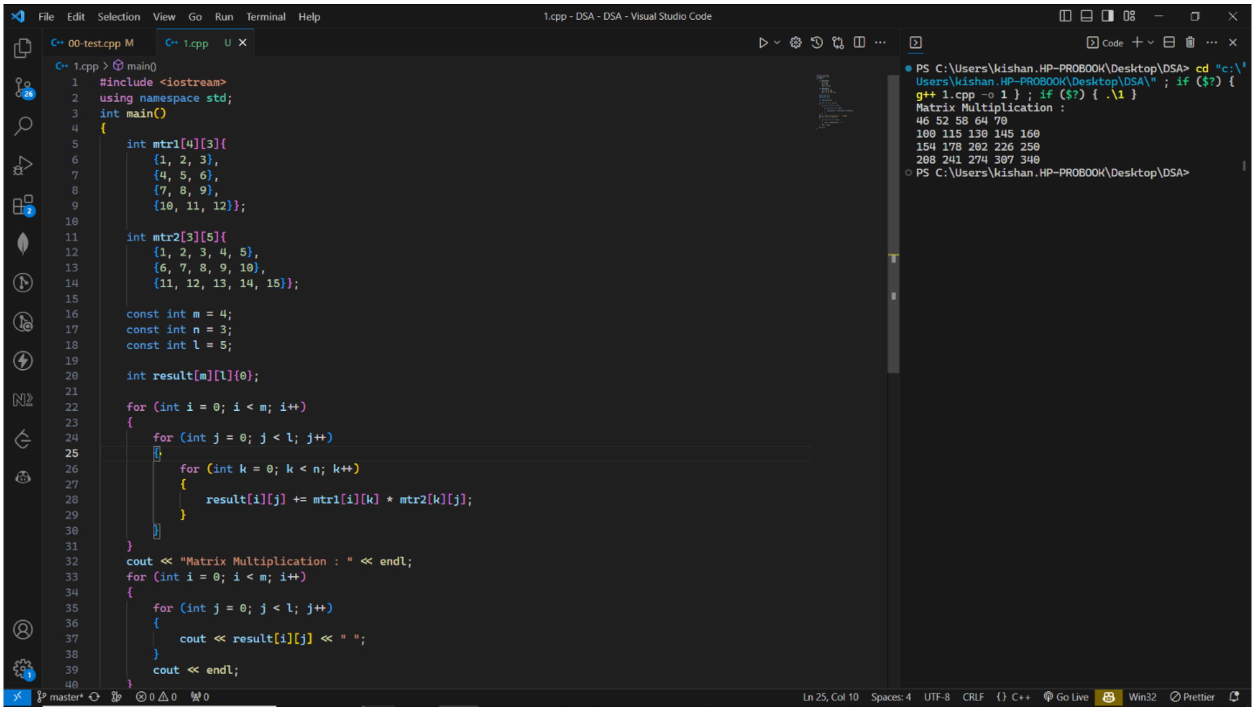
Student ID – 202312014

1)



```
#include <iostream>
using namespace std;
int main()
{
    int mtr1[4][3]{
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9},
        {10, 11, 12}};

    int mtr2[3][5]{
        {1, 2, 3, 4, 5},
        {6, 7, 8, 9, 10},
        {11, 12, 13, 14, 15}};

    const int m = 4;
    const int n = 3;
    const int l = 5;

    int result[m][l]{0};

    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < l; j++)
        {
            for (int k = 0; k < n; k++)
            {
                result[i][j] += mtr1[i][k] * mtr2[k][j];
            }
        }
    }
    cout << "Matrix Multiplication : " << endl;
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < l; j++)
        {
            cout << result[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

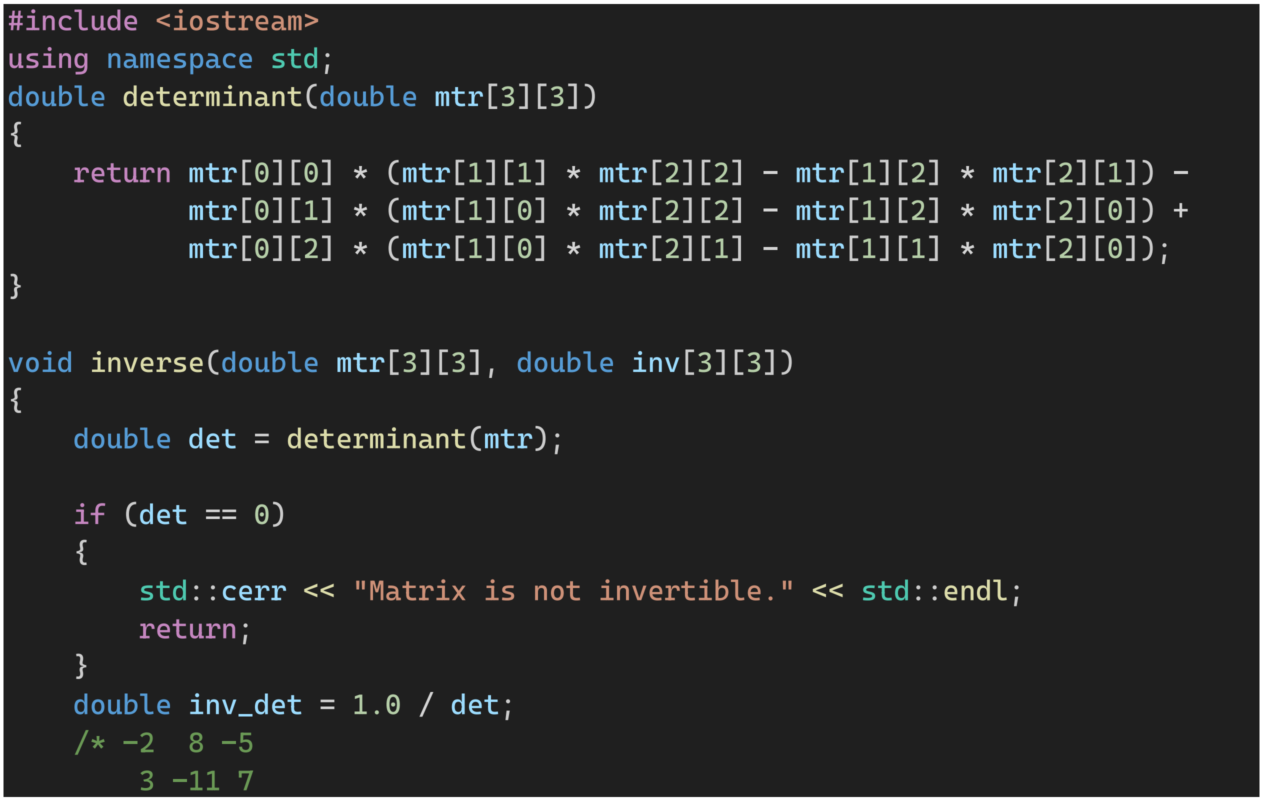
Output:



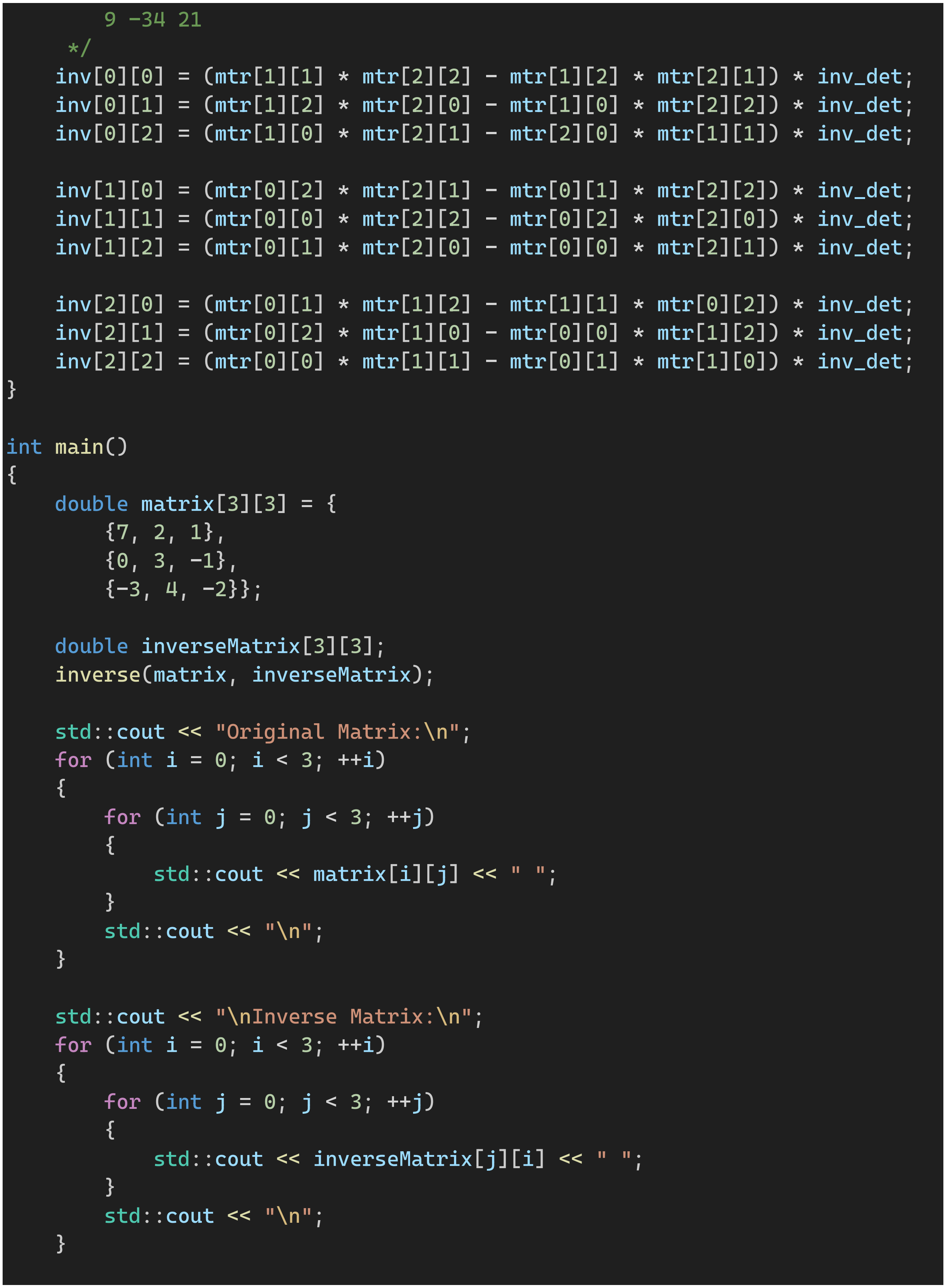
```cpp
#include <iostream>
using namespace std;
int main()
{
    int mtr1[4][3]{
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9},
        {10, 11, 12}};

    int mtr2[3][5]{
        {1, 2, 3, 4, 5},
        {6, 7, 8, 9, 10},
        {11, 12, 13, 14, 15}};

    const int m = 4;
    const int n = 3;
    const int l = 5;

    int result[m][l]{0};

    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < l; j++)
        {
            for (int k = 0; k < n; k++)
            {
                result[i][j] += mtr1[i][k] * mtr2[k][j];
            }
        }
    }
    cout << "Matrix Multiplication : " << endl;
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < l; j++)
        {
            cout << result[i][j] << " ";
        }
        cout << endl;
    }
```

```
PS C:\Users\kishan.HP-PROBOOK\Desktop\DSA> cd "c:\Users\kishan.HP-PROBOOK\Desktop\DSA\" ; if ($?) { g++ 1.cpp -o 1 } ; if ($?) { .\1 }
Matrix Multiplication :
46 52 58 64 70
100 115 130 145 160
154 178 202 226 250
208 241 274 307 340
PS C:\Users\kishan.HP-PROBOOK\Desktop\DSA>
```

2)

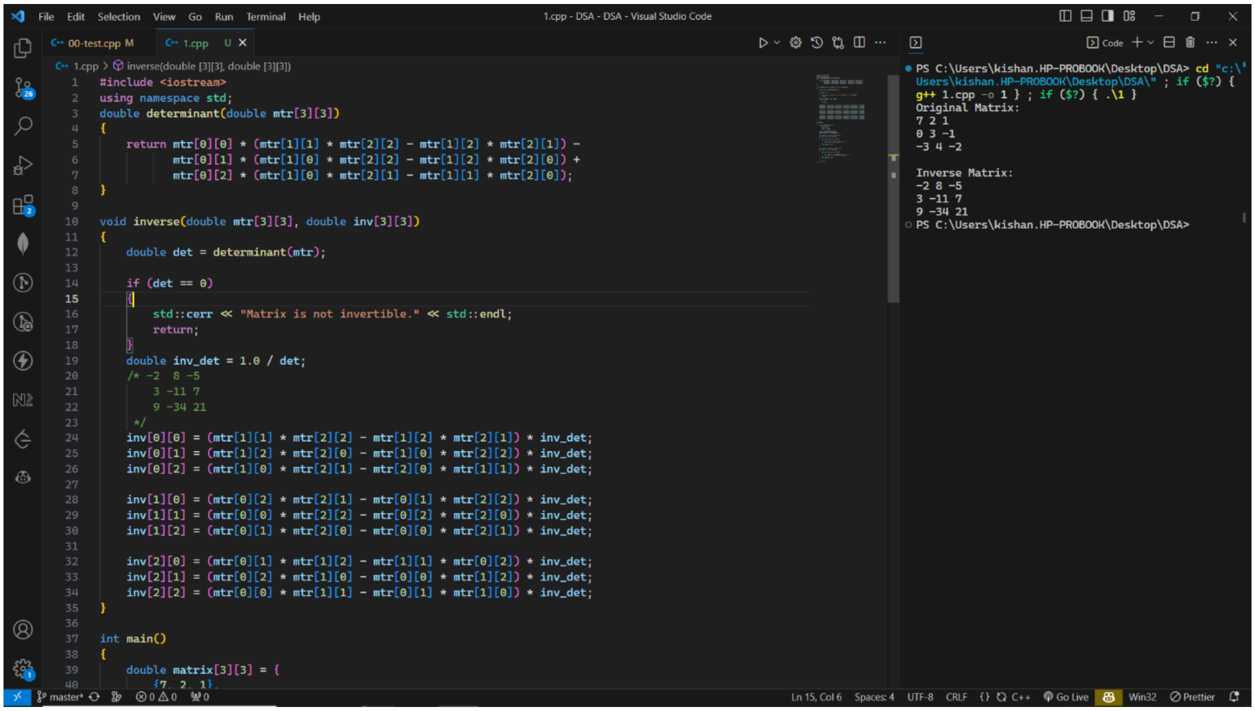


```cpp
#include <iostream>
using namespace std;
double determinant(double mtr[3][3])
{
    return mtr[0][0] * (mtr[1][1] * mtr[2][2] - mtr[1][2] * mtr[2][1]) -
           mtr[0][1] * (mtr[1][0] * mtr[2][2] - mtr[1][2] * mtr[2][0]) +
           mtr[0][2] * (mtr[1][0] * mtr[2][1] - mtr[1][1] * mtr[2][0]);
}

void inverse(double mtr[3][3], double inv[3][3])
{
    double det = determinant(mtr);

    if (det == 0)
    {
        std::cerr << "Matrix is not invertible." << std::endl;
        return;
    }
    double inv_det = 1.0 / det;
    /* -2  8 -5
        3 -11 7
```



```
        9 -34 21
     */
    inv[0][0] = (mtr[1][1] * mtr[2][2] - mtr[1][2] * mtr[2][1]) * inv_det;
    inv[0][1] = (mtr[1][2] * mtr[2][0] - mtr[1][0] * mtr[2][2]) * inv_det;
    inv[0][2] = (mtr[1][0] * mtr[2][1] - mtr[2][0] * mtr[1][1]) * inv_det;

    inv[1][0] = (mtr[0][2] * mtr[2][1] - mtr[0][1] * mtr[2][2]) * inv_det;
    inv[1][1] = (mtr[0][0] * mtr[2][2] - mtr[0][2] * mtr[2][0]) * inv_det;
    inv[1][2] = (mtr[0][1] * mtr[2][0] - mtr[0][0] * mtr[2][1]) * inv_det;

    inv[2][0] = (mtr[0][1] * mtr[1][2] - mtr[1][1] * mtr[0][2]) * inv_det;
    inv[2][1] = (mtr[0][2] * mtr[1][0] - mtr[0][0] * mtr[1][2]) * inv_det;
    inv[2][2] = (mtr[0][0] * mtr[1][1] - mtr[0][1] * mtr[1][0]) * inv_det;
}

int main()
{
    double matrix[3][3] = {
        {7, 2, 1},
        {0, 3, -1},
        {-3, 4, -2}};

    double inverseMatrix[3][3];
    inverse(matrix, inverseMatrix);

    std::cout << "Original Matrix:\n";
    for (int i = 0; i < 3; ++i)
    {
        for (int j = 0; j < 3; ++j)
        {
            std::cout << matrix[i][j] << " ";
        }
        std::cout << "\n";
    }

    std::cout << "\nInverse Matrix:\n";
    for (int i = 0; i < 3; ++i)
    {
        for (int j = 0; j < 3; ++j)
        {
            std::cout << inverseMatrix[j][i] << " ";
        }
        std::cout << "\n";
    }
    return 0;
}
```

Output:


File Edit Selection View Go Run Terminal Help
1.cpp - DSA - DSA - Visual Studio Code
00-test.cpp M
1.cpp U
1.cpp > inverse(double [3][3], double [3][3])
Code
master*
Ln 15, Col 6 Spaces: 4 UTF-8 CRLF {} C++ Go Live Win32 Prettier

```
#include <iostream>
using namespace std;
double determinant(double mtr[3][3])
{
    return mtr[0][0] * (mtr[1][1] * mtr[2][2] - mtr[1][2] * mtr[2][1]) -
           mtr[0][1] * (mtr[1][0] * mtr[2][2] - mtr[1][2] * mtr[2][0]) +
           mtr[0][2] * (mtr[1][0] * mtr[2][1] - mtr[1][1] * mtr[2][0]);
}

void inverse(double mtr[3][3], double inv[3][3])
{
    double det = determinant(mtr);

    if (det == 0)
    {
        std::cerr << "Matrix is not invertible." << std::endl;
        return;
    }
    double inv_det = 1.0 / det;
    /* -2  8 -5
        3 -11 7
        9 -34 21
     */
    inv[0][0] = (mtr[1][1] * mtr[2][2] - mtr[1][2] * mtr[2][1]) * inv_det;
    inv[0][1] = (mtr[1][2] * mtr[2][0] - mtr[1][0] * mtr[2][2]) * inv_det;
    inv[0][2] = (mtr[1][0] * mtr[2][1] - mtr[2][0] * mtr[1][1]) * inv_det;

    inv[1][0] = (mtr[0][2] * mtr[2][1] - mtr[0][1] * mtr[2][2]) * inv_det;
    inv[1][1] = (mtr[0][0] * mtr[2][2] - mtr[0][2] * mtr[2][0]) * inv_det;
    inv[1][2] = (mtr[0][1] * mtr[2][0] - mtr[0][0] * mtr[2][1]) * inv_det;

    inv[2][0] = (mtr[0][1] * mtr[1][2] - mtr[1][1] * mtr[0][2]) * inv_det;
    inv[2][1] = (mtr[0][2] * mtr[1][0] - mtr[0][0] * mtr[1][2]) * inv_det;
    inv[2][2] = (mtr[0][0] * mtr[1][1] - mtr[0][1] * mtr[1][0]) * inv_det;
}

int main()
{
    double matrix[3][3] = {
        {7, 2, 1},
```

```
PS C:\Users\kishan.HP-PROBOOK\Desktop\DSA> cd "c:\Users\kishan.HP-PROBOOK\Desktop\DSA\" ; if ($?) { g++ 1.cpp -o 1 } ; if ($?) { .\1 }
Original Matrix:
7 2 1
0 3 -1
-3 4 -2

Inverse Matrix:
-2 8 -5
3 -11 7
9 -34 21
PS C:\Users\kishan.HP-PROBOOK\Desktop\DSA>
```

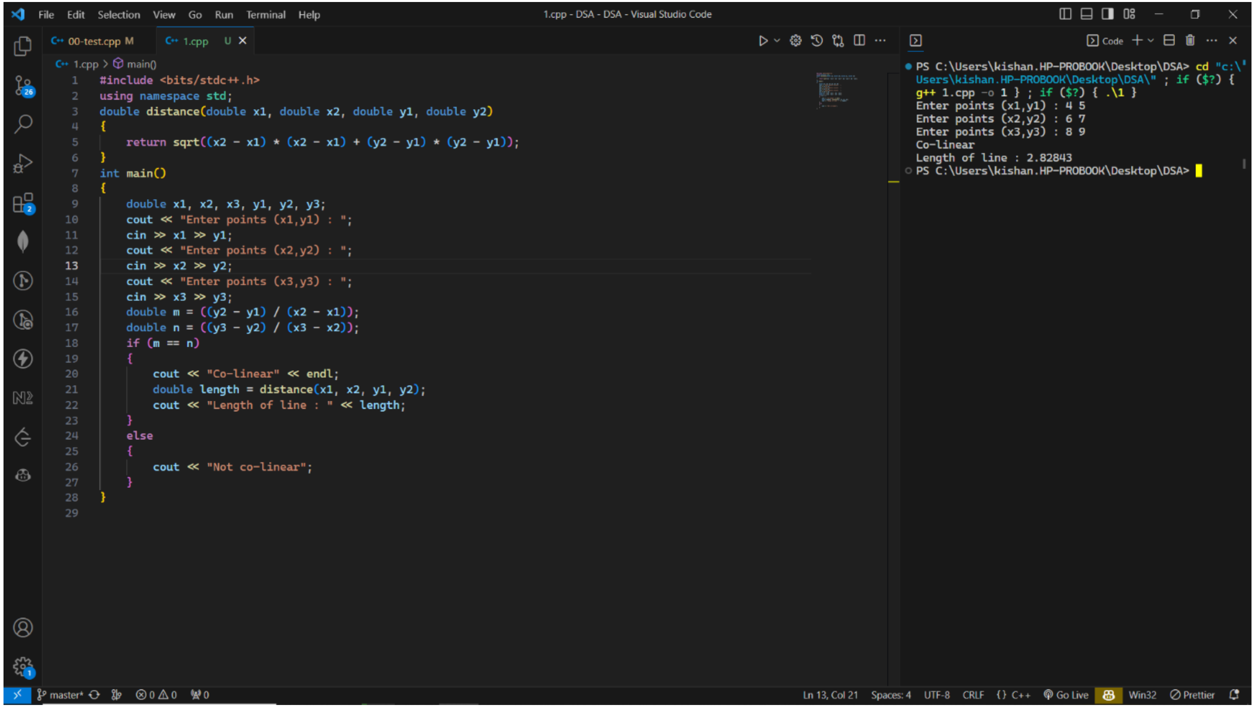
3)

i) if they are colinear, if so, find the length of the line

```
#include <bits/stdc++.h>
using namespace std;
double distance(double x1, double x2, double y1, double y2)
{
    return sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
}
int main()
{
    double x1, x2, x3, y1, y2, y3;
    cout << "Enter points (x1,y1) : ";
    cin >> x1 >> y1;
    cout << "Enter points (x2,y2) : ";
    cin >> x2 >> y2;
    cout << "Enter points (x3,y3) : ";
    cin >> x3 >> y3;
    double m = ((y2 - y1) / (x2 - x1));
```

```cpp
    double n = ((y3 - y2) / (x3 - x2));
    if (m == n)
    {
        cout << "Co-linear" << endl;
        double length = distance(x1, x2, y1, y2);
        cout << "Length of line : " << length;
    }
    else
    {
        cout << "Not co-linear";
    }
}
```

Output:


1.cpp - DSA - DSA - Visual Studio Code
File Edit Selection View Go Run Terminal Help
00-test.cpp M
1.cpp U
1.cpp > main()
Code
master*
Ln 13, Col 21 Spaces: 4 UTF-8 CRLF {} C++ Go Live Win32 Prettier

```cpp
#include <bits/stdc++.h>
using namespace std;
double distance(double x1, double x2, double y1, double y2)
{
    return sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
}
int main()
{
    double x1, x2, x3, y1, y2, y3;
    cout << "Enter points (x1,y1) : ";
    cin >> x1 >> y1;
    cout << "Enter points (x2,y2) : ";
    cin >> x2 >> y2;
    cout << "Enter points (x3,y3) : ";
    cin >> x3 >> y3;
    double m = ((y2 - y1) / (x2 - x1));
    double n = ((y3 - y2) / (x3 - x2));
    if (m == n)
    {
        cout << "Co-linear" << endl;
        double length = distance(x1, x2, y1, y2);
        cout << "Length of line : " << length;
    }
    else
    {
        cout << "Not co-linear";
    }
}
```

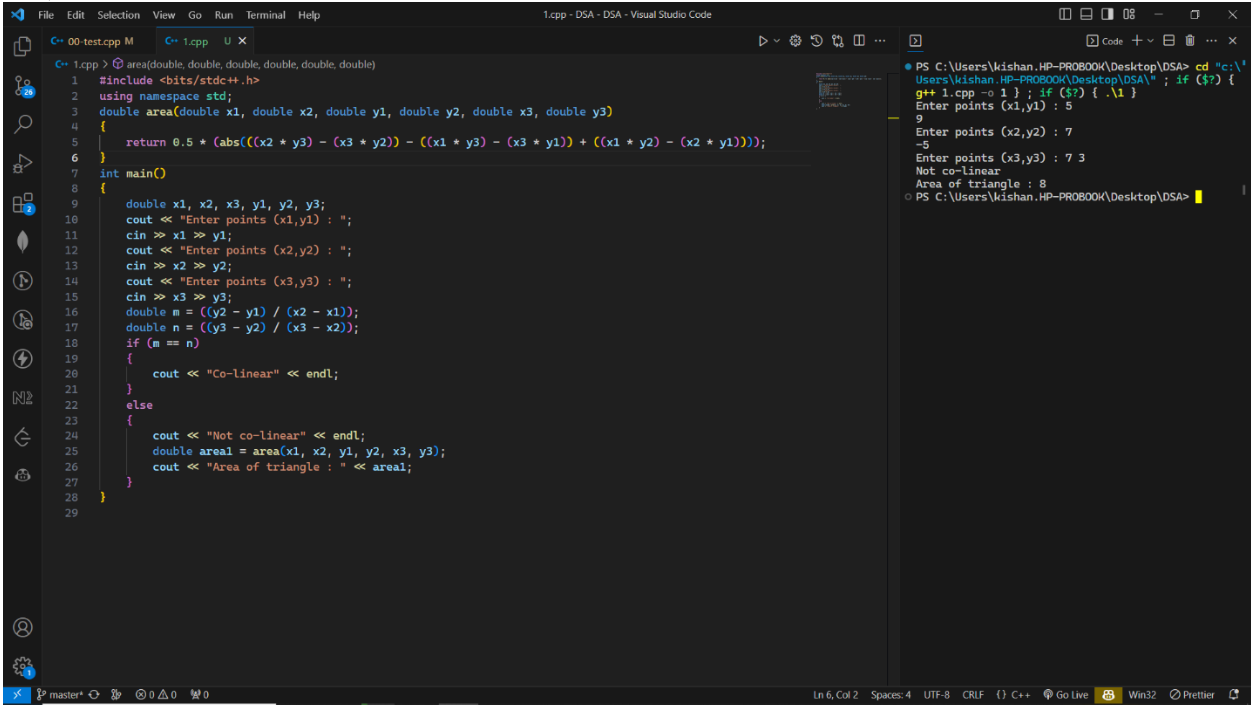
```
PS C:\Users\kishan.HP-PROBOOK\Desktop\DSA> cd "c:\Users\kishan.HP-PROBOOK\Desktop\DSA\" ; if ($?) { g++ 1.cpp -o 1 } ; if ($?) { .\1 }
Enter points (x1,y1) : 4 5
Enter points (x2,y2) : 6 7
Enter points (x3,y3) : 8 9
Co-linear
Length of line : 2.82843
PS C:\Users\kishan.HP-PROBOOK\Desktop\DSA>
```

ii) if they are not colinear, if so find the area of triangle using determinant method

```cpp
#include <bits/stdc++.h>
using namespace std;
double area(double x1, double x2, double y1, double y2, double x3, double
y3)
{
    return 0.5 * (abs(((x2 * y3) - (x3 * y2)) - ((x1 * y3) - (x3 * y1)) +
((x1 * y2) - (x2 * y1))));
}
int main()
```

```cpp
{
    double x1, x2, x3, y1, y2, y3;
    cout << "Enter points (x1,y1) : ";
    cin >> x1 >> y1;
    cout << "Enter points (x2,y2) : ";
    cin >> x2 >> y2;
    cout << "Enter points (x3,y3) : ";
    cin >> x3 >> y3;
    double m = ((y2 - y1) / (x2 - x1));
    double n = ((y3 - y2) / (x3 - x2));
    if (m == n)
    {
        cout << "Co-linear" << endl;
    }
    else
    {
        cout << "Not co-linear" << endl;
        double area1 = area(x1, x2, y1, y2, x3, y3);
        cout << "Area of triangle : " << area1;
    }
}
```

Output:



```cpp
#include <bits/stdc++.h>
using namespace std;
double area(double x1, double x2, double y1, double y2, double x3, double y3)
{
    return 0.5 * (abs(((x2 * y3) - (x3 * y2)) - ((x1 * y3) - (x3 * y1)) + ((x1 * y2) - (x2 * y1))));
}
int main()
{
    double x1, x2, x3, y1, y2, y3;
    cout << "Enter points (x1,y1) : ";
    cin >> x1 >> y1;
    cout << "Enter points (x2,y2) : ";
    cin >> x2 >> y2;
    cout << "Enter points (x3,y3) : ";
    cin >> x3 >> y3;
    double m = ((y2 - y1) / (x2 - x1));
    double n = ((y3 - y2) / (x3 - x2));
    if (m == n)
    {
        cout << "Co-linear" << endl;
    }
    else
    {
        cout << "Not co-linear" << endl;
        double area1 = area(x1, x2, y1, y2, x3, y3);
        cout << "Area of triangle : " << area1;
    }
}
```

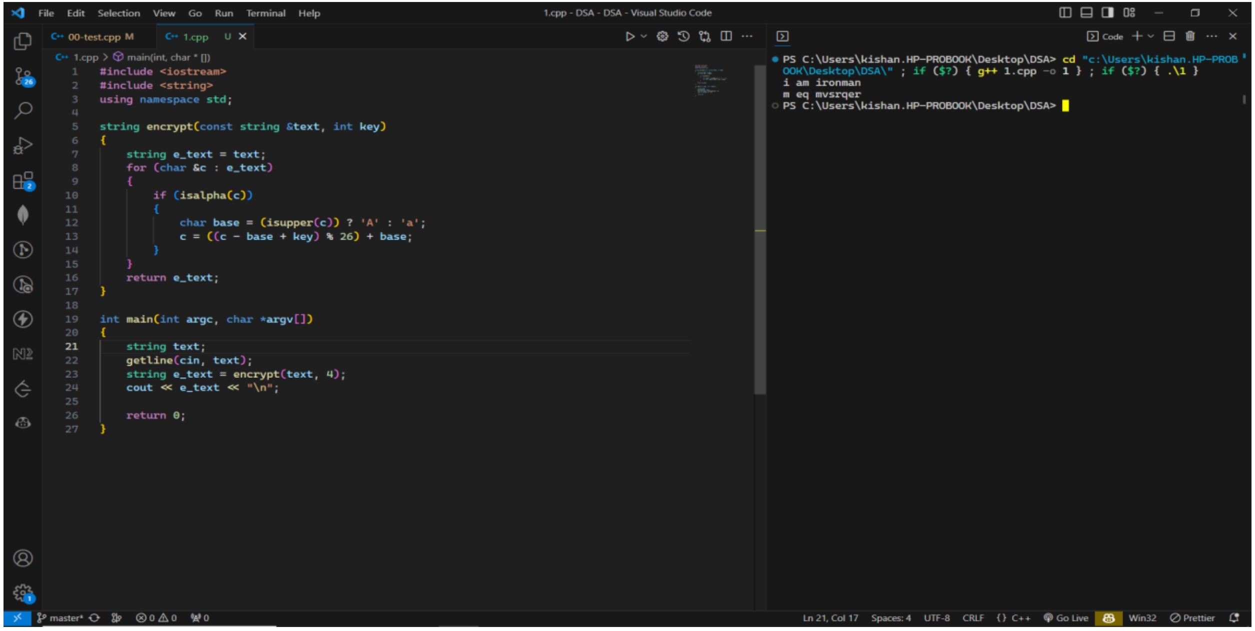
```
PS C:\Users\kishan.HP-PROBOOK\Desktop\DSA> cd "c:\Users\kishan.HP-PROBOOK\Desktop\DSA\" ; if ($?) { g++ 1.cpp -o 1 } ; if ($?) { .\1 }
Enter points (x1,y1) : 5
9
Enter points (x2,y2) : 7
-5
Enter points (x3,y3) : 7 3
Not co-linear
Area of triangle : 8
PS C:\Users\kishan.HP-PROBOOK\Desktop\DSA>
```

4)

For Encryption

```cpp
#include <iostream>
#include <string>
using namespace std;

string encrypt(const string &text, int key)
{
    string e_text = text;
    for (char &c : e_text)
    {
        if (isalpha(c))
        {
            char base = (isupper(c)) ? 'A' : 'a';
            c = ((c - base + key) % 26) + base;
        }
    }
    return e_text;
}

int main(int argc, char *argv[])
{
    string text;
    getline(cin, text);
    string e_text = encrypt(text, 4);
    cout << e_text << "\n";
    return 0;
}
```

Output:


File Edit Selection View Go Run Terminal Help
1.cpp - DSA - DSA - Visual Studio Code
00-test.cpp M
1.cpp U
1.cpp > main(int, char * [])
Code
master*
0 0
0
Ln 21, Col 17 Spaces: 4 UTF-8 CRLF {} C++ Go Live Win32 Prettier

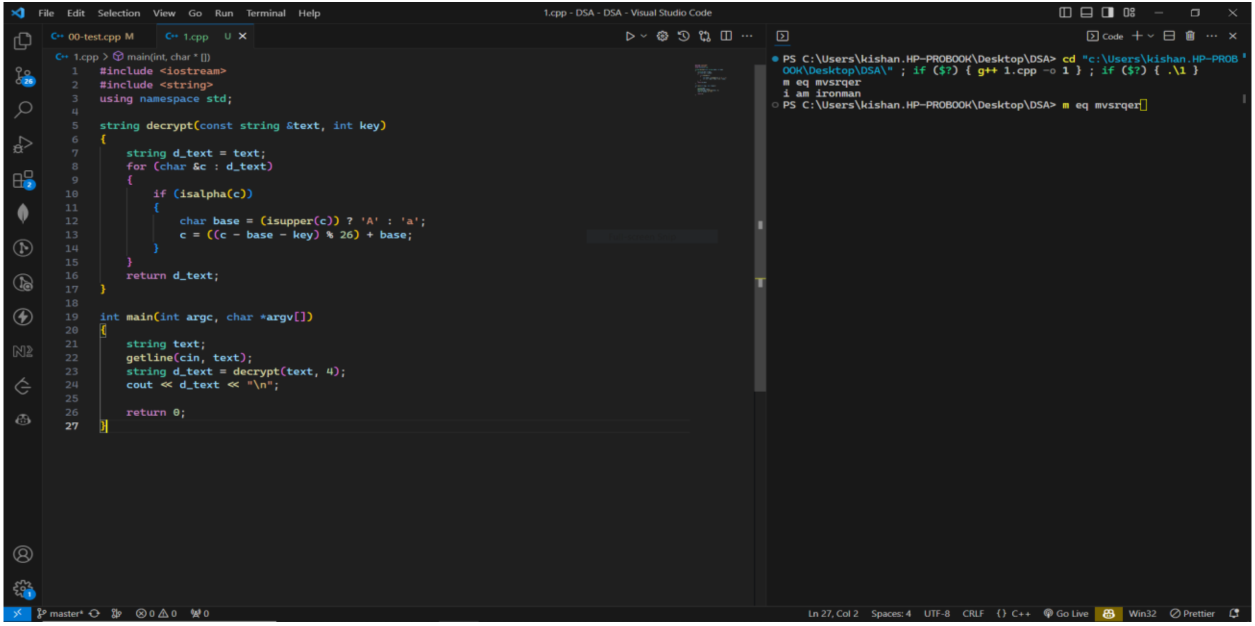
```cpp
#include <iostream>
#include <string>
using namespace std;

string encrypt(const string &text, int key)
{
    string e_text = text;
    for (char &c : e_text)
    {
        if (isalpha(c))
        {
            char base = (isupper(c)) ? 'A' : 'a';
            c = ((c - base + key) % 26) + base;
        }
    }
    return e_text;
}

int main(int argc, char *argv[])
{
    string text;
    getline(cin, text);
    string e_text = encrypt(text, 4);
    cout << e_text << "\n";

    return 0;
}
```

```
PS C:\Users\kishan.HP-PROBOOK\Desktop\DSA> cd "c:\Users\kishan.HP-PROBOOK\Desktop\DSA\" ; if ($?) { g++ 1.cpp -o 1 } ; if ($?) { .\1 }
i am ironman
m eq mvsrqer
PS C:\Users\kishan.HP-PROBOOK\Desktop\DSA>
```

For decryption

```cpp
#include <iostream>
#include <string>
using namespace std;

string decrypt(const string &text, int key)
{
    string d_text = text;
    for (char &c : d_text)
    {
        if (isalpha(c))
        {
            char base = (isupper(c)) ? 'A' : 'a';
            c = ((c - base - key) % 26) + base;
        }
    }
    return d_text;
}

int main(int argc, char *argv[])
{
    string text;
    getline(cin, text);
    string d_text = decrypt(text, 4);
    cout << d_text << "\n";
    return 0;
}
```

Output

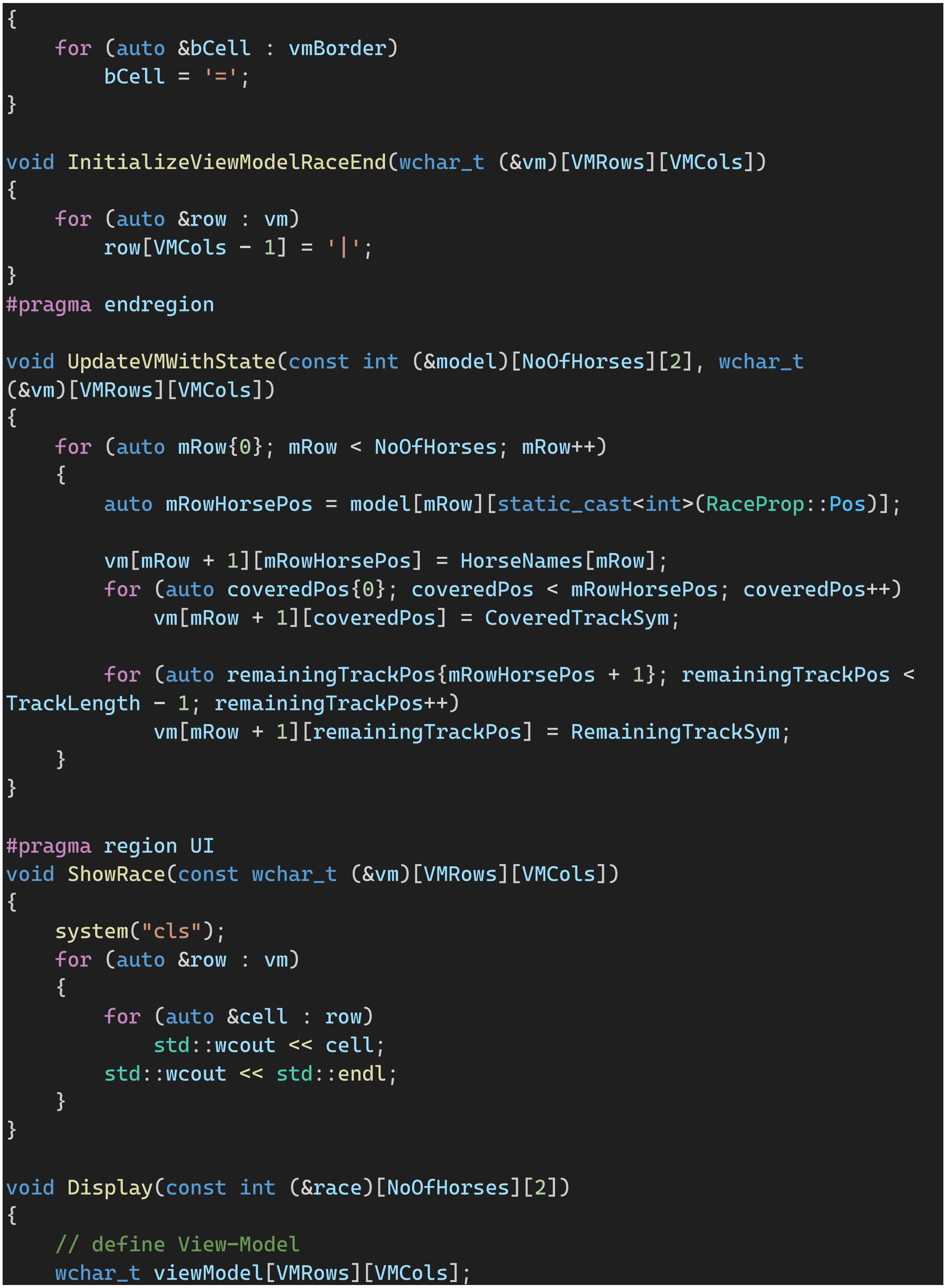

File Edit Selection View Go Run Terminal Help
1.cpp - DSA - DSA - Visual Studio Code
00-test.cpp M
1.cpp U
1.cpp > main(int, char * [])
Code
master*
0 0
0
Ln 27, Col 2 Spaces: 4 UTF-8 CRLF {} C++ Go Live Win32 Prettier

```cpp
#include <iostream>
#include <string>
using namespace std;

string decrypt(const string &text, int key)
{
    string d_text = text;
    for (char &c : d_text)
    {
        if (isalpha(c))
        {
            char base = (isupper(c)) ? 'A' : 'a';
            c = ((c - base - key) % 26) + base;
        }
    }
    return d_text;
}

int main(int argc, char *argv[])
{
    string text;
    getline(cin, text);
    string d_text = decrypt(text, 4);
    cout << d_text << "\n";

    return 0;
}
```

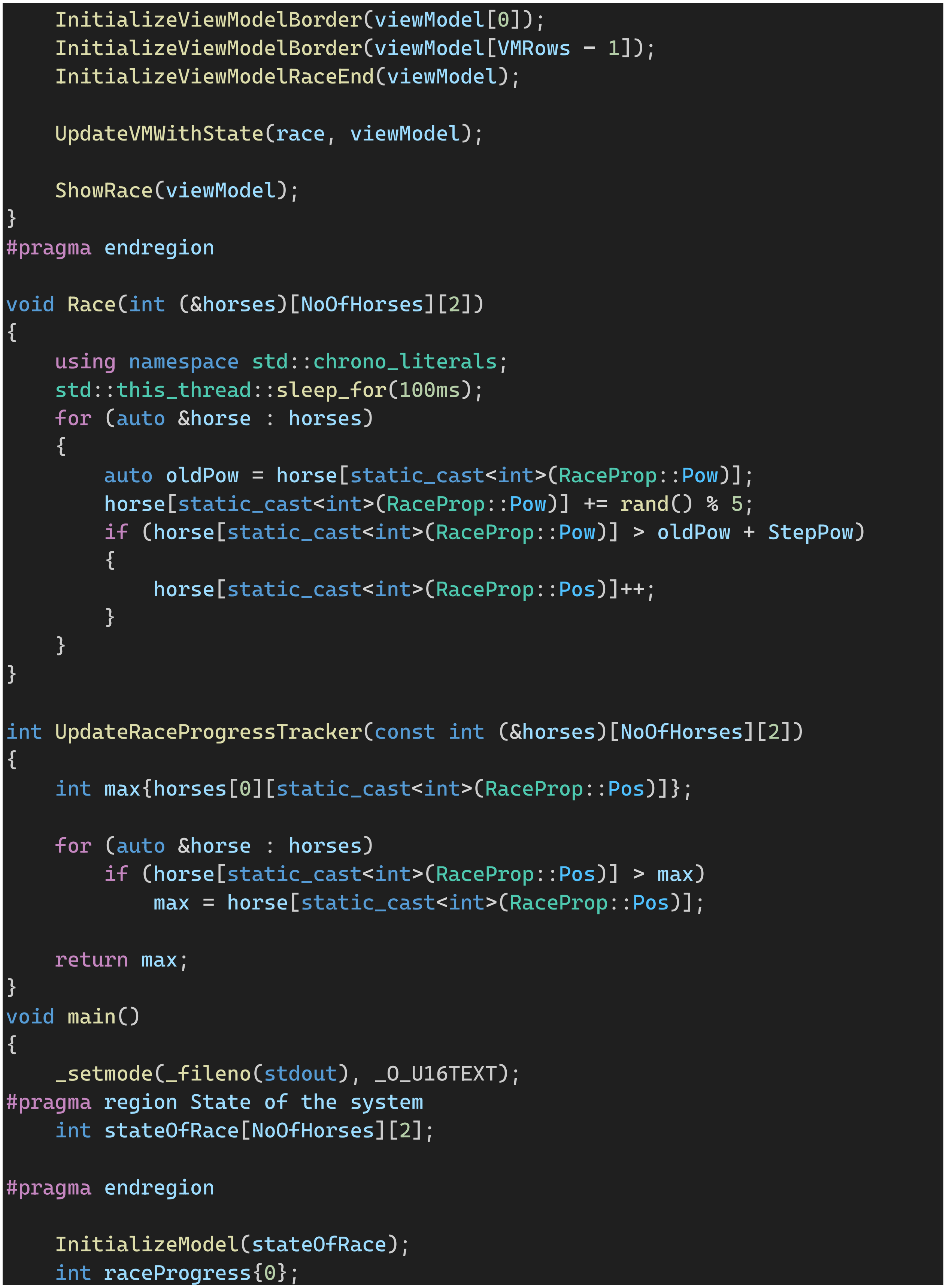
```
PS C:\Users\kishan.HP-PROBOOK\Desktop\DSA> cd "c:\Users\kishan.HP-PROBOOK\Desktop\DSA\" ; if ($?) { g++ 1.cpp -o 1 } ; if ($?) { .\1 }
m eq mvsrqer
i am ironman
PS C:\Users\kishan.HP-PROBOOK\Desktop\DSA> m eq mvsrqer
```

5)

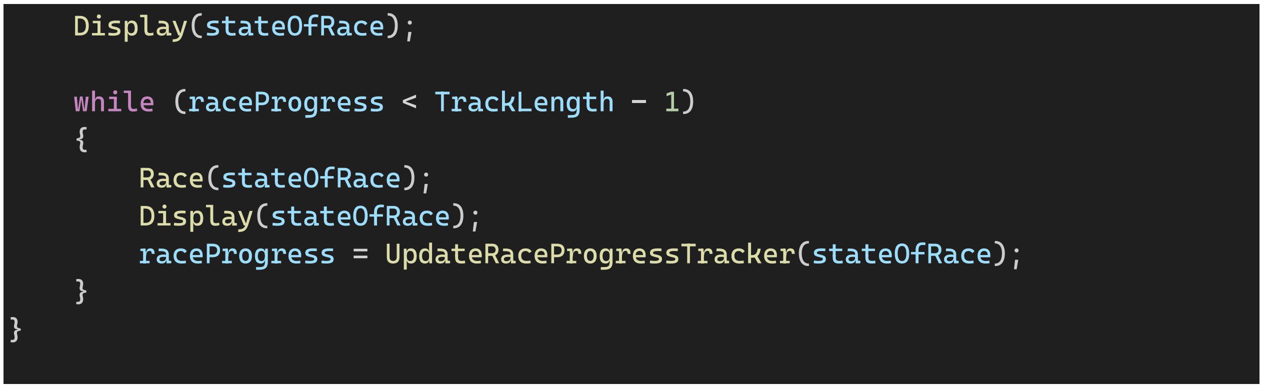
```
#include <chrono>
#include <thread>
#include <bits/stdc++.h>
#include <io.h>
#include <fcntl.h>
#pragma region Consts
enum class RaceProp
{
    Pos,
    Pow
};
const int NoOfHorses{4};
const int TrackLength{50};
const int VMRows{NoOfHorses + 2};
const int VMCols{TrackLength};
const wchar_t HorseNames[]{'X', 'M', 'T', 'D'};
const wchar_t CoveredTrackSym{'-'};
const wchar_t RemainingTrackSym{'.'};
const int StepPow{3};
#pragma endregion

#pragma region Init
void InitializeModel(int (&horses)[NoOfHorses][2])
{
    for (auto &row : horses)
        for (auto &col : row)
            col = 0;

    for (auto &row : horses)
    {
        row[static_cast<int>(RaceProp::Pos)] = 0;
        row[static_cast<int>(RaceProp::Pow)] = 0;
    }

    for (int i{0}; i < NoOfHorses; i++)
        horses[i][static_cast<int>(RaceProp::Pos)] = 0;

    for (int i{0}; i < NoOfHorses; i++)
        horses[i][static_cast<int>(RaceProp::Pow)] = 0;
}

void InitializeViewModelBorder(wchar_t (&vmBorder)[VMCols])
```



```
{
    for (auto &bCell : vmBorder)
        bCell = '=';
}

void InitializeViewModelRaceEnd(wchar_t (&vm)[VMRows][VMCols])
{
    for (auto &row : vm)
        row[VMCols - 1] = '|';
}
#pragma endregion

void UpdateVMWithState(const int (&model)[NoOfHorses][2], wchar_t
(&vm)[VMRows][VMCols])
{
    for (auto mRow{0}; mRow < NoOfHorses; mRow++)
    {
        auto mRowHorsePos = model[mRow][static_cast<int>(RaceProp::Pos)];

        vm[mRow + 1][mRowHorsePos] = HorseNames[mRow];
        for (auto coveredPos{0}; coveredPos < mRowHorsePos; coveredPos++)
            vm[mRow + 1][coveredPos] = CoveredTrackSym;

        for (auto remainingTrackPos{mRowHorsePos + 1}; remainingTrackPos <
TrackLength - 1; remainingTrackPos++)
            vm[mRow + 1][remainingTrackPos] = RemainingTrackSym;
    }
}

#pragma region UI
void ShowRace(const wchar_t (&vm)[VMRows][VMCols])
{
    system("cls");
    for (auto &row : vm)
    {
        for (auto &cell : row)
            std::wcout << cell;
        std::wcout << std::endl;
    }
}

void Display(const int (&race)[NoOfHorses][2])
{
    // define View-Model
    wchar_t viewModel[VMRows][VMCols];
```

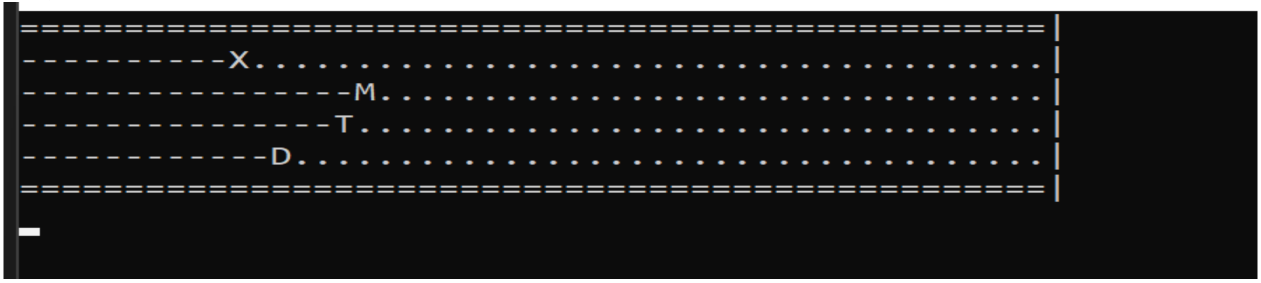


```
    InitializeViewModelBorder(viewModel[0]);
    InitializeViewModelBorder(viewModel[VMRows - 1]);
    InitializeViewModelRaceEnd(viewModel);

    UpdateVMWithState(race, viewModel);

    ShowRace(viewModel);
}
#pragma endregion

void Race(int (&horses)[NoOfHorses][2])
{
    using namespace std::chrono_literals;
    std::this_thread::sleep_for(100ms);
    for (auto &horse : horses)
    {
        auto oldPow = horse[static_cast<int>(RaceProp::Pow)];
        horse[static_cast<int>(RaceProp::Pow)] += rand() % 5;
        if (horse[static_cast<int>(RaceProp::Pow)] > oldPow + StepPow)
        {
            horse[static_cast<int>(RaceProp::Pos)]++;
        }
    }
}

int UpdateRaceProgressTracker(const int (&horses)[NoOfHorses][2])
{
    int max{horses[0][static_cast<int>(RaceProp::Pos)]};

    for (auto &horse : horses)
        if (horse[static_cast<int>(RaceProp::Pos)] > max)
            max = horse[static_cast<int>(RaceProp::Pos)];

    return max;
}
void main()
{
    _setmode(_fileno(stdout), _O_U16TEXT);
#pragma region State of the system
    int stateOfRace[NoOfHorses][2];

#pragma endregion

    InitializeModel(stateOfRace);
    int raceProgress{0};
```

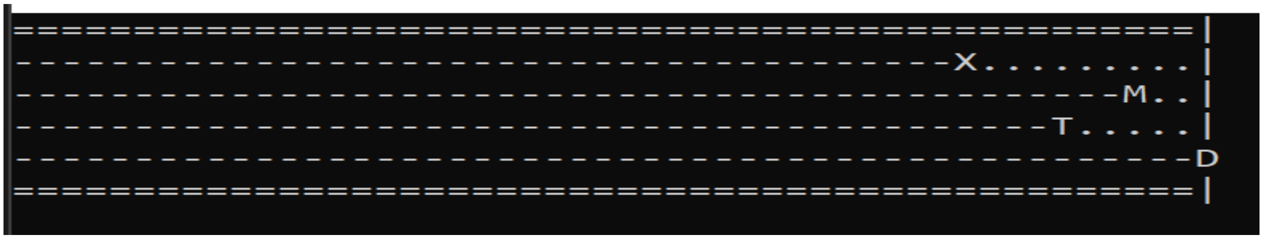


```
    Display(stateOfRace);

    while (raceProgress < TrackLength - 1)
    {
        Race(stateOfRace);
        Display(stateOfRace);
        raceProgress = UpdateRaceProgressTracker(stateOfRace);
    }
}
```

Output:



```
=================================================|
----------X......................................|
----------------M................................|
---------------T.................................|
------------D....................................|
=================================================|
```



```
=================================================|
---------------------------------------X.........|
----------------------------------------------M..|
-------------------------------------------T.....|
-------------------------------------------------D
=================================================|
```