

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}};

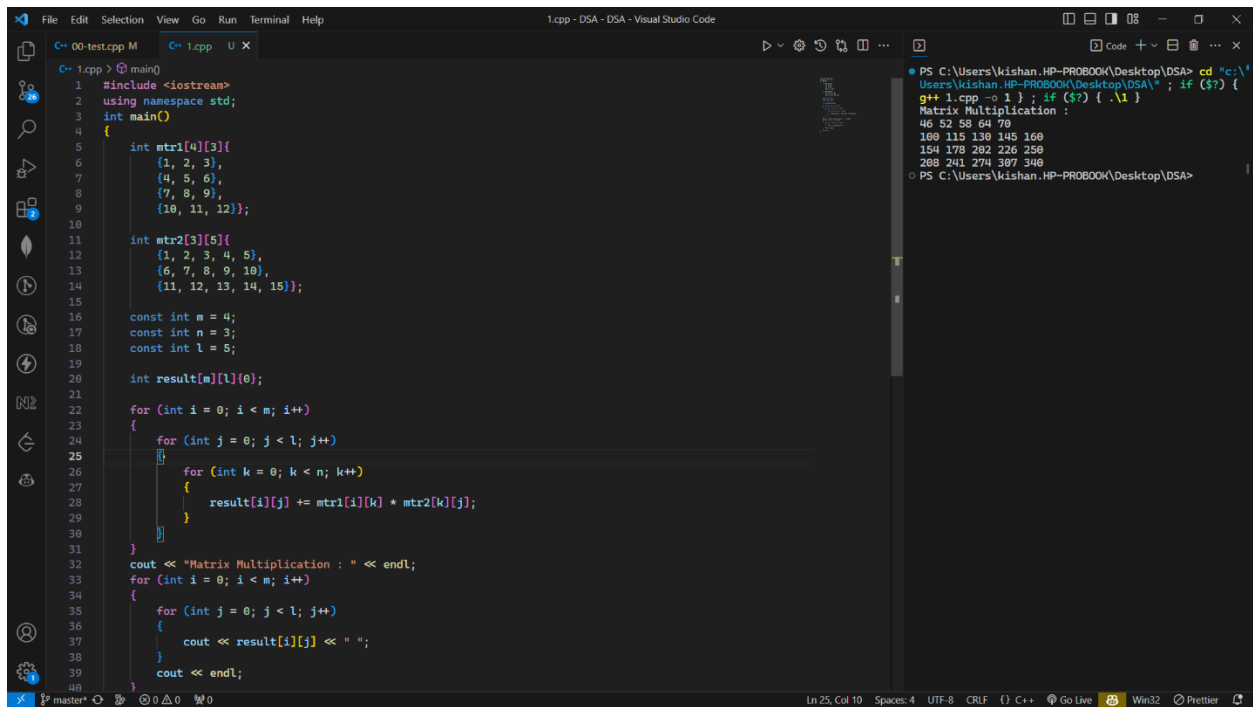
    const int m = 4;
    const int n = 3;
    const int l = 5;

    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:



```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int mtr1[4][3]{
6         {1, 2, 3},
7         {4, 5, 6},
8         {7, 8, 9},
9         {10, 11, 12}};
10
11     int mtr2[3][5]{
12         {1, 2, 3, 4, 5},
13         {6, 7, 8, 9, 10},
14         {11, 12, 13, 14, 15}};
15
16     const int m = 4;
17     const int n = 3;
18     const int l = 5;
19
20     int result[m][l]{0};
21
22     for (int i = 0; i < m; i++)
23     {
24         for (int j = 0; j < l; j++)
25         {
26             for (int k = 0; k < n; k++)
27             {
28                 result[i][j] += mtr1[i][k] * mtr2[k][j];
29             }
30         }
31     }
32     cout << "Matrix Multiplication : " << endl;
33     for (int i = 0; i < m; i++)
34     {
35         for (int j = 0; j < l; j++)
36         {
37             cout << result[i][j] << " ";
38         }
39         cout << endl;
40     }
```

```
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)

```
#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:

The screenshot shows a Visual Studio Code editor with a C++ file named '1.cpp'. The code implements a function to calculate the determinant of a 3x3 matrix and another function to calculate its inverse. The main function initializes a matrix and calls the inverse function. The terminal window on the right shows the output of the program, which prints the original matrix and its inverse.

```

1  #include <iostream>
2  using namespace std;
3  double determinant(double mtr[3][3])
4  {
5      return mtr[0][0] * (mtr[1][1] * mtr[2][2] - mtr[1][2] * mtr[2][1]) -
6             mtr[0][1] * (mtr[1][0] * mtr[2][2] - mtr[1][2] * mtr[2][0]) +
7             mtr[0][2] * (mtr[1][0] * mtr[2][1] - mtr[1][1] * mtr[2][0]);
8  }
9
10 void inverse(double mtr[3][3], double inv[3][3])
11 {
12     double det = determinant(mtr);
13
14     if (det == 0)
15     {
16         std::cerr << "Matrix is not invertible." << std::endl;
17         return;
18     }
19     double inv_det = 1.0 / det;
20     /* -2  8 -5
21        3 -11 7
22        9 -34 21
23     */
24     inv[0][0] = (mtr[1][1] * mtr[2][2] - mtr[1][2] * mtr[2][1]) * inv_det;
25     inv[0][1] = (mtr[1][2] * mtr[2][0] - mtr[1][0] * mtr[2][2]) * inv_det;
26     inv[0][2] = (mtr[1][0] * mtr[2][1] - mtr[2][0] * mtr[1][1]) * inv_det;
27
28     inv[1][0] = (mtr[0][2] * mtr[2][1] - mtr[0][1] * mtr[2][2]) * inv_det;
29     inv[1][1] = (mtr[0][0] * mtr[2][2] - mtr[0][2] * mtr[2][0]) * inv_det;
30     inv[1][2] = (mtr[0][1] * mtr[2][0] - mtr[0][0] * mtr[2][1]) * inv_det;
31
32     inv[2][0] = (mtr[0][1] * mtr[1][2] - mtr[0][1] * mtr[0][2]) * inv_det;
33     inv[2][1] = (mtr[0][2] * mtr[1][0] - mtr[0][0] * mtr[1][2]) * inv_det;
34     inv[2][2] = (mtr[0][0] * mtr[1][1] - mtr[0][1] * mtr[1][0]) * inv_det;
35 }
36
37 int main()
38 {
39     double matrix[3][3] = {
40         {7, 2, 1},

```

```

PS C:\Users\kishan.HP-PROBOOK\Desktop\DSA> cd "c:\Users\kishan.HP-PROBOOK\Desktop\DSA" && g++ 1.cpp -o 1 && if ($?) {
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));

```

```

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 #include <bits/stdc++.h>
2 using namespace std;
3 double distance(double x1, double x2, double y1, double y2)
4 {
5     return sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
6 }
7 int main()
8 {
9     double x1, x2, x3, y1, y2, y3;
10    cout << "Enter points (x1,y1) : ";
11    cin >> x1 >> y1;
12    cout << "Enter points (x2,y2) : ";
13    cin >> x2 >> y2;
14    cout << "Enter points (x3,y3) : ";
15    cin >> x3 >> y3;
16    double m = ((y2 - y1) / (x2 - x1));
17    double n = ((y3 - y2) / (x3 - x2));
18    if (m == n)
19    {
20        cout << "Co-linear" << endl;
21        double length = distance(x1, x2, y1, y2);
22        cout << "Length of line : " << length;
23    }
24    else
25    {
26        cout << "Not co-linear";
27    }
28 }
29

```

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

```

#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;
    }
}

```

Output:

The screenshot shows the Visual Studio Code editor with a C++ file named '1.cpp'. The code defines a function 'area' that calculates the area of a triangle given three points (x1, y1), (x2, y2), and (x3, y3). The function uses the formula: $0.5 * (abs((x2 * y3) - (x3 * y2)) - ((x1 * y3) - (x3 * y1)) + ((x1 * y2) - (x2 * y1)))$. The main function prompts the user to enter three points and checks if they are co-linear. If they are, it prints 'Co-linear'; otherwise, it prints 'Not co-linear' and the area of the triangle.

The terminal output shows the following sequence of events:

```

PS C:\Users\kishan.HP-PROBOOK\Desktop\DSA> cd "c:\Users\kishan.HP-PROBOOK\Desktop\DSA"
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)

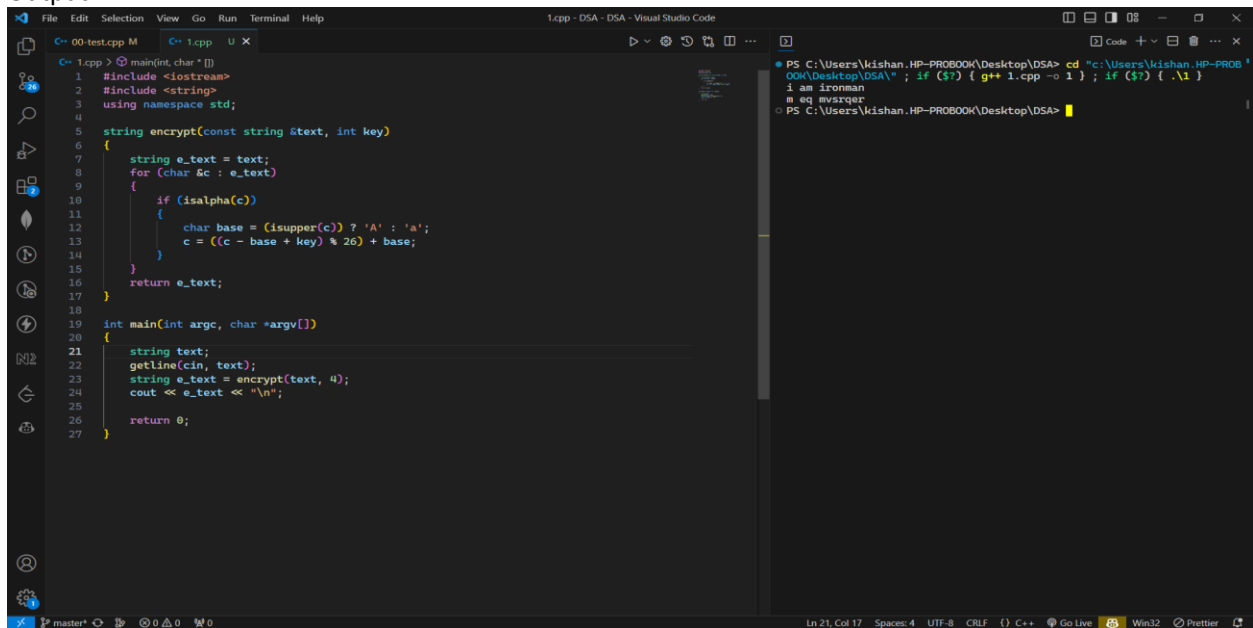
For Encryption

```
#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:



The screenshot shows the Visual Studio Code editor with a C++ file named '1.cpp'. The code is the same as shown in the previous block. The output window on the right shows the command prompt output: 'I am Ironman' followed by a newline, resulting in 'I am Ironman' on one line and 'a en avssqer' on the next line. The status bar at the bottom indicates the file is at line 21, column 17, with 4 spaces, UTF-8 encoding, CRLF line endings, and C++ language.

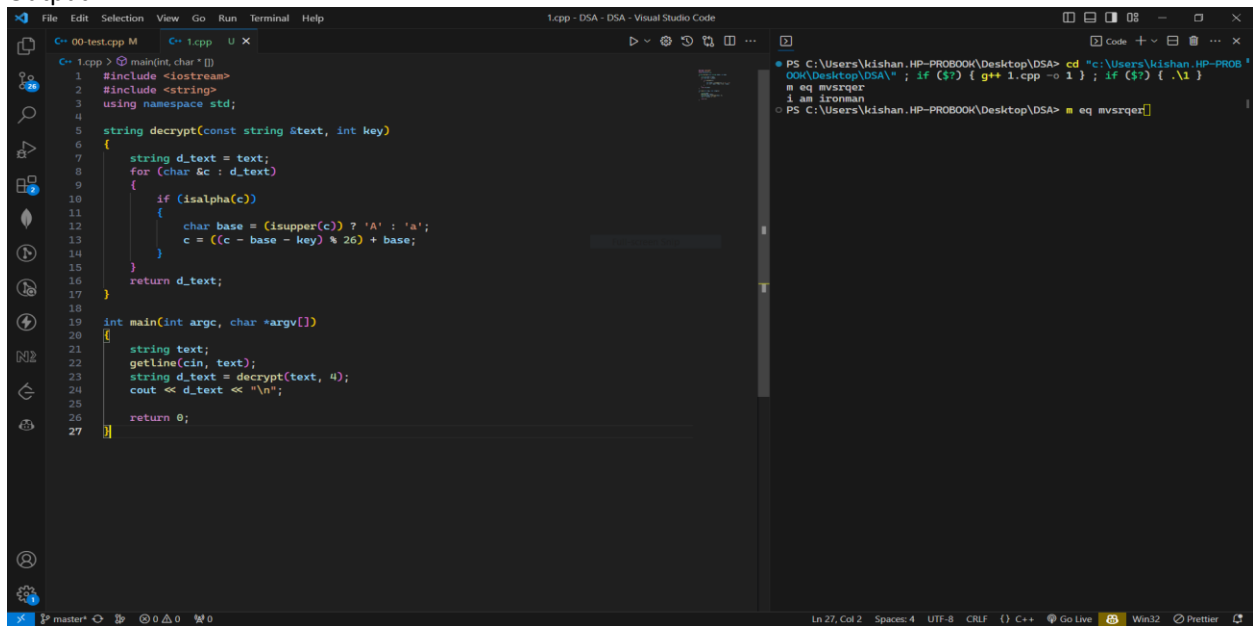
For decryption

```
#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

The screenshot shows the Visual Studio Code editor with a C++ file named '1.cpp'. The code is the same as shown in the previous block. The output window on the right shows the command prompt results: 'PS C:\Users\Kishan.HP-PROBOOK\Desktop\DSA> cd "c:\Users\Kishan.HP-PROBOOK\Desktop\DSA\" ; if (\$?) { g++ 1.cpp -o 1 } ; if (\$?) { .\1 }' followed by the input 'I am Ironman' and the output 'I am Ironman'. The status bar at the bottom indicates the file is on the 'master' branch, line 27, column 2, with 0 errors and 0 warnings. The editor is using C++ with the Win32 target and the Prettier formatter.

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:

