Securin Assessment

Problem Statement: The Doomed Dice Challenge

Part-A

1. How many total combinations are possible?

Math:

Total number of combinations when rolling two dice together will be obtained by multiplying the number of faces on Die A by the number of faces on Die B.

Total combinations = Number of faces on Die A \* Number of faces on Die B

both Die A and Die B have 6 faces each so, total combinations = 6 \* 6 = 36

2. Calculate and display the distribution of all possible combinations that can be obtained when rolling both Die A and Die B together.

Math:

Die A and Die B has 6 faces numbered from 1 to 6.

Possible pair of combinations are obtained by taking each face of Die A and combining it with each face of Die B.

A 6 x 6 matrix is required and each cell in the matrix represents a combination of faces from Die A and Die B, forming pairs like (1, 1), (1, 2), (1, 3), ..., (6, 5), (6, 6).

First, we take the first face of Die A i.e., 1 and combine it with all faces of Die B (1 to 6).

And then the second face of Die A, combine it with all faces of Die B (1 to 6).

This process continues until we combine each face of Die A with each face of Die B.

Die A: [1, 2, 3, 4, 5, 6]

Die B: [1, 2, 3, 4, 5, 6]

The matrix of combinations:

(1, 1) (1, 2) (1, 3) (1, 4) (1, 5) (1, 6)

(2, 1) (2, 2) (2, 3) (2, 4) (2, 5) (2, 6)

(3, 1) (3, 2) (3, 3) (3, 4) (3, 5) (3, 6)

(4, 1) (4, 2) (4, 3) (4, 4) (4, 5) (4, 6)

(5, 1) (5, 2) (5, 3) (5, 4) (5, 5) (5, 6)

(6, 1) (6, 2) (6, 3) (6, 4) (6, 5) (6, 6)

3. Calculate the Probability of all Possible Sums occurring among the number of combinations from (2).

The probability of each possible sum occurring is calculated by counting the occurrences of each sum in the matrix of combinations and dividing the count by the total number of combinations.

Sum = 2:

There's only one combination where the sum is 2:(1,1).

Number of combinations resulting in a sum of 2 = 1.

Total number of combinations = 36.

Probability of occurrence of sum 2 (P (Sum = 2)) = Number of combinations resulting in a sum of 2 / Total number of combinations = 1 / 36.

Sum = 3:

Occurrences = 2 ((1,2) and (2,1))

Probability = Occurrences / Total combinations = 2 / 36

Sum = 4:

Occurrences = 3 ((1,3), (2,2), (3,1))

Probability = 3 / 36

Sum = 5:

Occurrences = 4 ((1,4), (2,3), (3,2), (4,1))

Probability = 4 / 36

Sum = 6:

Occurrences = 5 ((1,5), (2,4), (3,3), (4,2), (5,1))

Probability = 5 / 36

Sum = 7:

Occurrences = 6 ((1,6), (2,5), (3,4), (4,3), (5,2), (6,1))

Probability = 6 / 36

Sum = 8:

Occurrences = 5 ((2,6), (3,5), (4,4), (5,3), (6,2))

Probability = 5 / 36

Sum = 9:

Occurrences = 4 ((3,6), (4,5), (5,4), (6,3))

Probability = 4 / 36

Sum = 10:

Occurrences = 3 ((4,6), (5,5), (6,4))

Probability = 3 / 36

Sum = 11:

Occurrences = 2 ((5,6), (6,5))

Probability = 2 / 36

Sum = 12:

Occurrences = 1 (6,6)

Probability = 1 / 36

Code:

#include <iostream>

#include <vector>

using namespace std;

int main(){

vector<int> Die\_A = {1, 2, 3, 4, 5, 6};

vector<int> Die\_B = {1, 2, 3, 4, 5, 6};

int total\_combinations = Die\_A.size() \* Die\_B.size();

cout << "Total combinations: " << total\_combinations <<endl;

vector<vector<pair<int, int>>> combinations(6, vector<pair<int, int>>(6));

vector<int> sum\_count(13);

for (int i = 0; i < 6; i++) {

for (int j = 0; j < 6; j++) {

pair<int, int> combination = make\_pair(Die\_A[i], Die\_B[j]);

combinations[i][j] = combination;

int sum = Die\_A[i] + Die\_B[j];

sum\_count[sum]++;

}

}

// Display the 6x6 matrix of combinations

cout << "\nMatrix of combinations:\n";

for (int i = 0; i < 6; i++) {

for (int j = 0; j < 6; j++) {

pair<int, int> combination = combinations[i][j];

cout << "(" << combination.first << ", " << combination.second << ") ";

}

cout << endl;

}

// Display probabilities for each sum

cout << "\nProbability of each sum:\n";

for (int i = 2; i <= 12; i++) {

double probability = static\_cast<double>(sum\_count[i]) / total\_combinations;

cout << "P(Sum = " << i << ") = " << sum\_count[i] << "/36 = " << probability << endl;

}

return 0;

}

Code Logic:

Two vectors `Die\_A` and `Die\_B` represent the faces of the dice.

A 6x6 matrix named `combinations` is defined to store all possible combinations of rolling Die A and Die B together.

Another vector `sum\_count` of size 13 is defined to count occurrences for sums from 2 to 12 (since two dice summing to 2 or 12 is the minimum and maximum possible sums).

Matrix Combinations:

Nested loops iterate through each face of Die A and Die B, create pairs, and stores them in the `combinations` matrix.

For each combination, it calculates the sum of faces from Die A and Die B and updates the count of occurrences for that sum in the sum\_count vector.

After storing combinations in the matrix, it displays the 6 x 6 matrix.

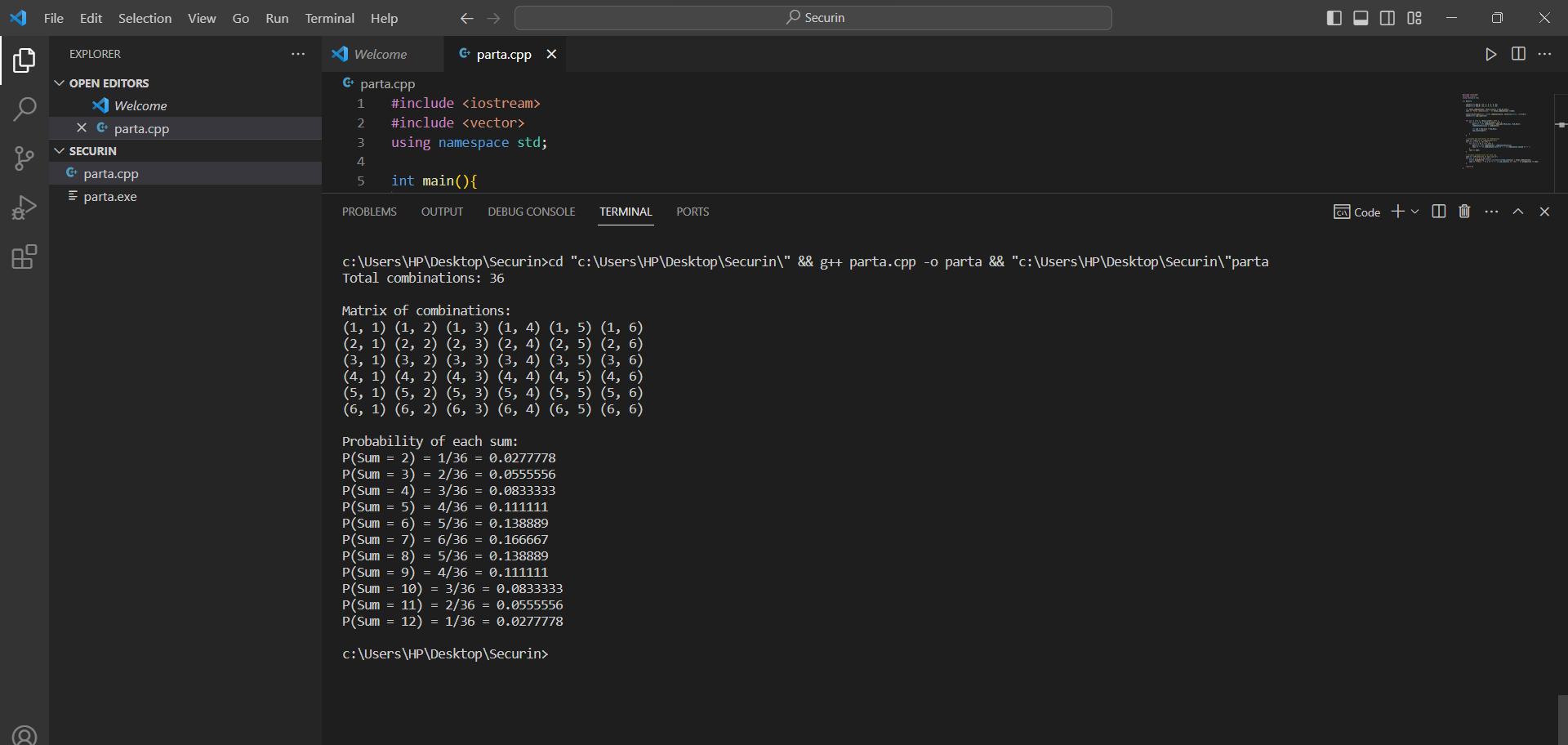
Calculating Probabilities:

Using the sum\_count vector, it calculates the probability of each possible sum from 2 to 12 occurring when rolling two dice.

It traverses through the sum\_count vector and divides the count of each sum by the total number of combinations (36) to obtain the probability.

The probabilities for each sum are then displayed.

Result:



Part-B

Code:

#include <iostream>

#include <vector>

using namespace std;

void diceACombo(vector<int>& elements, int length, vector<int>& current, vector<vector<int>>& allCombinations) {

if (current.size() == length) {

allCombinations.push\_back(current);

return;

}

for (int element : elements) {

current.push\_back(element);

diceACombo(elements, length, current, allCombinations);

current.pop\_back();

}

}

void diceBCombo(vector<int>& elements, int length, int start, vector<int>& current, vector<vector<int>>& allCombinations) {

if (current.size() == length) {

allCombinations.push\_back(current);

return;

}

for (int i = start; i < elements.size(); i++) {

current.push\_back(elements[i]);

diceBCombo(elements, length, i + 1, current, allCombinations);

current.pop\_back();

}

}

vector<double> probSum(const vector<int>& arr1, const vector<int>& arr2) {

vector<double> psum1(12, 0.0);

for (int i : arr1) {

for (int j : arr2) {

int k = i + j;

psum1[k - 1] += 1;

}

}

for (int i = 0; i < psum1.size(); i++) {

if (psum1[i] != 0) {

psum1[i] /= 36;

}

}

return psum1;

}

void transform(const vector<int>& dieA, const vector<int>& dieB) {

vector<int> elements1 = {1, 2, 3, 4};

int length = 6;

vector<int> current;

vector<vector<int>> combo1;

diceACombo(elements1, length, current, combo1);

vector<int> elements2 = {1, 2, 3, 4, 5, 6, 7, 8};

int start = 0;

vector<vector<int>> combo2;

diceBCombo(elements2, length, start, current, combo2);

vector<double> psum = {0.0, 1.0 / 36, 2.0 / 36, 3.0 / 36, 4.0 / 36, 5.0 / 36, 6.0 / 36, 5.0 / 36, 4.0 / 36, 3.0 / 36, 2.0 / 36, 1.0 / 36};

bool flag = false;

for (const auto& i : combo1) {

for (const auto& j : combo2) {

if (probSum(i, j) == psum) {

cout << "new die\_a: ";

for (int x : i) {

cout << x << " ";

}

cout << "\nnew die\_b: ";

for (int x : j) {

cout << x << " ";

}

cout << endl;

flag = true;

break;

}

}

if (flag) {

break;

}

}

}

int main() {

vector<int> dieA = {1, 2, 3, 4, 5, 6};

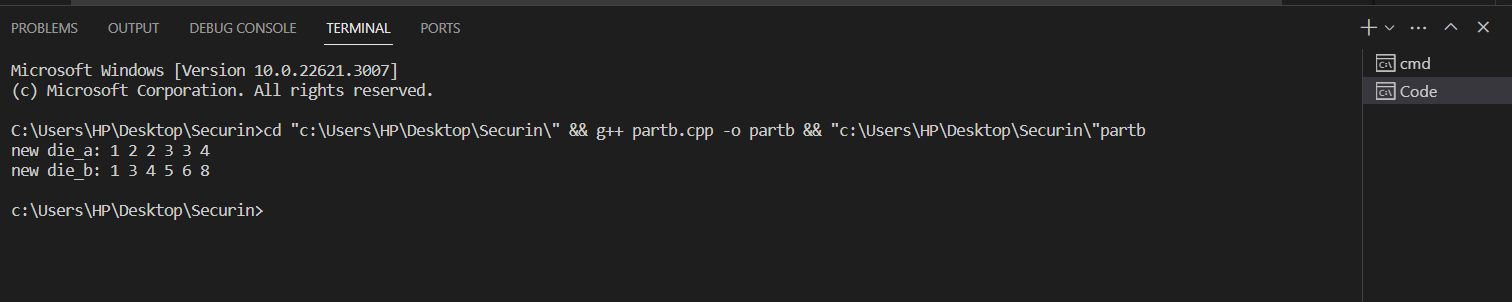
vector<int> dieB = {1, 2, 3, 4, 5, 6};

transform(dieA, dieB);

return 0;

}

Result:



Github Link:

[Jeevana1609/Dice (github.com)](https://github.com/Jeevana1609/Dice)