



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

DOMAIN WINTER WINNING CAMP(Day-8)

Student Name:Shreyansh Vishmoi

Branch: BE-CSE

Semester: 6th

UID: 22BCS15373

Section/Group: 620-B

Date of Performance: 29/12/24

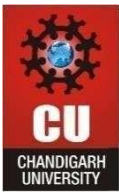
1) N-th Tribonacci Number

```
#include <iostream>
#include <vector>
using namespace std;

int tribonacci(int n) {
    vector<int> dp(max(3, n + 1));
    dp[0] = 0;
    dp[1] = dp[2] = 1;
    for (int i = 3; i <= n; i++) {
        dp[i] = dp[i - 1] + dp[i - 2] + dp[i - 3];
    }
    return dp[n];
}

int main() {
    int n = 4;
    cout << "Tribonacci(" << n << ") = " << tribonacci(n) << endl;
    return 0;
}
```

main.cpp	Run	Output
<pre>1 #include <iostream> 2 #include <vector> 3 using namespace std; 4 5 int tribonacci(int n) { 6 vector<int> dp(max(3, n + 1)); 7 dp[0] = 0;</pre>		<pre>Tribonacci(4) = 4 === Code Execution Suc</pre>



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

2) Divisor Game

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

bool divisorGame(int n) {
    return n % 2 == 0;
}

int main() {
    int n;
    cout << "Enter the value of n: ";
    cin >> n;
    cout << "Alice wins: " << (divisorGame(n) ? "Yes" : "No") << endl;
    return 0;
}
```

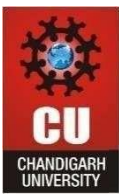
main.cpp	Run	Output
<pre>1 #include <iostream> 2 using namespace std; 3 4 bool divisorGame(int n) { 5 return n % 2 == 0; 6 }</pre>		<pre>Enter the value of n: 5 Alice wins: No === Code Execution Success</pre>

3) Climbing Stairs

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

int climbStairs(int n) {
    if (n <= 2) return n;
    int a = 1, b = 2;
    for (int i = 3; i <= n; i++) {
        int temp = a + b;
        a = b;
        b = temp;
    }
    return b;
}

int main() {
    int n;
    cout << "Enter the number of steps: ";
    cin >> n;
    cout << "Ways to climb " << n << " stairs: " << climbStairs(n) << endl;
    return 0;
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Output :

```
main.cpp  [Icons]  Run  Output
1 #include <iostream>
2 using namespace std;
3
4 int climbStairs(int n) {
5     if (n <= 2) return n;
6     int a = 1, b = 2;
7     for (int i = 3; i <= n; i++) {
8         int temp = a + b;
9         a = b;
10        b = temp;
11    }
12    return b;
13 }
```

Enter the number of steps: 2
Ways to climb 2 stairs: 2
=== Code Execution Successful

4) Best Time to Buy and Sell Stock

```
#include <iostream>
#include <vector>
#include <climits> // For INT_MAX
using namespace std;

int maxProfit(vector<int>& prices) {
    int minPrice = INT_MAX, maxProfit = 0;
    for (int price : prices) {
        minPrice = min(minPrice, price); // Update the minimum price seen so far
        maxProfit = max(maxProfit, price - minPrice); // Calculate the maximum profit
    }
    return maxProfit;
}

int main() {
    int n;
    cout << "Enter the number of days: ";
    cin >> n;

    vector<int> prices(n);
    cout << "Enter the stock prices: ";
    for (int i = 0; i < n; i++) {
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
cin >> prices[i];  
}
```

```
cout << "Max profit: " << maxProfit(prices) << endl;  
return 0;  
}
```

The screenshot displays a web browser with the Programiz online C++ compiler. The code in the editor is as follows:

```
1 #include <iostream>  
2 #include <vector>  
3 #include <limits> // For INT_MAX  
4 using namespace std;  
5  
6 int maxProfit(vector<int>& prices) {  
7     int minPrice = INT_MAX, maxProfit = 0;  
8     for (int price : prices) {  
9         minPrice = min(minPrice, price); // Update the minimum  
           price seen so far  
10        maxProfit = max(maxProfit, price - minPrice); //  
           Calculate the maximum profit  
11    }  
12    return maxProfit;  
13 }  
14  
15 int main() {  
16     int n;  
17     cout << "Enter the number of days: ";  
18     cin >> n;
```

The output window shows the following execution results:

```
Enter the number of days: 6  
Enter the stock prices: 7  
1  
5  
3  
6  
4  
Max profit: 5  
  
=== Code Execution Successful ===
```

The Windows taskbar at the bottom indicates the system temperature is 60°F and cloudy, with the date and time being 7:45 PM on 12/28/2024.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

5) Generate Parentheses

```
#include <iostream>
#include <vector>
using namespace std;

void generate(int open, int close, string current, vector<string>& result) {
    if (open == 0 && close == 0) {
        result.push_back(current);
        return;
    }
    if (open > 0) generate(open - 1, close, current + "(", result);
    if (close > open) generate(open, close - 1, current + ")", result);
}

vector<string> generateParenthesis(int n) {
    vector<string> result;
    generate(n, n, "", result);
    return result;
}

int main() {
    int n;
    cout << "Enter the number of pairs: ";
    cin >> n;
    vector<string> result = generateParenthesis(n);
    for (const string& s : result) {
        cout << s << endl;
    }
    return 0;
}
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
main.cpp
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 void generate(int open, int close, string current, vector
  <string>& result) {
6     if (open == 0 && close == 0) {
7         result.push_back(current);
8         return;
9     }
10    if (open > 0) generate(open - 1, close, current + "(",
        result);
11    if (close > open) generate(open, close - 1, current + ")",
        result);
12 }
13
14 vector<string> generateParenthesis(int n) {
15     vector<string> result;
16     generate(n, n, "", result);
17     return result;
```

Output

Enter the number of pairs: 3

((()))
(()())
()()()
()()()

=== Code Execution Successful ===

6) Minimum Path Sum

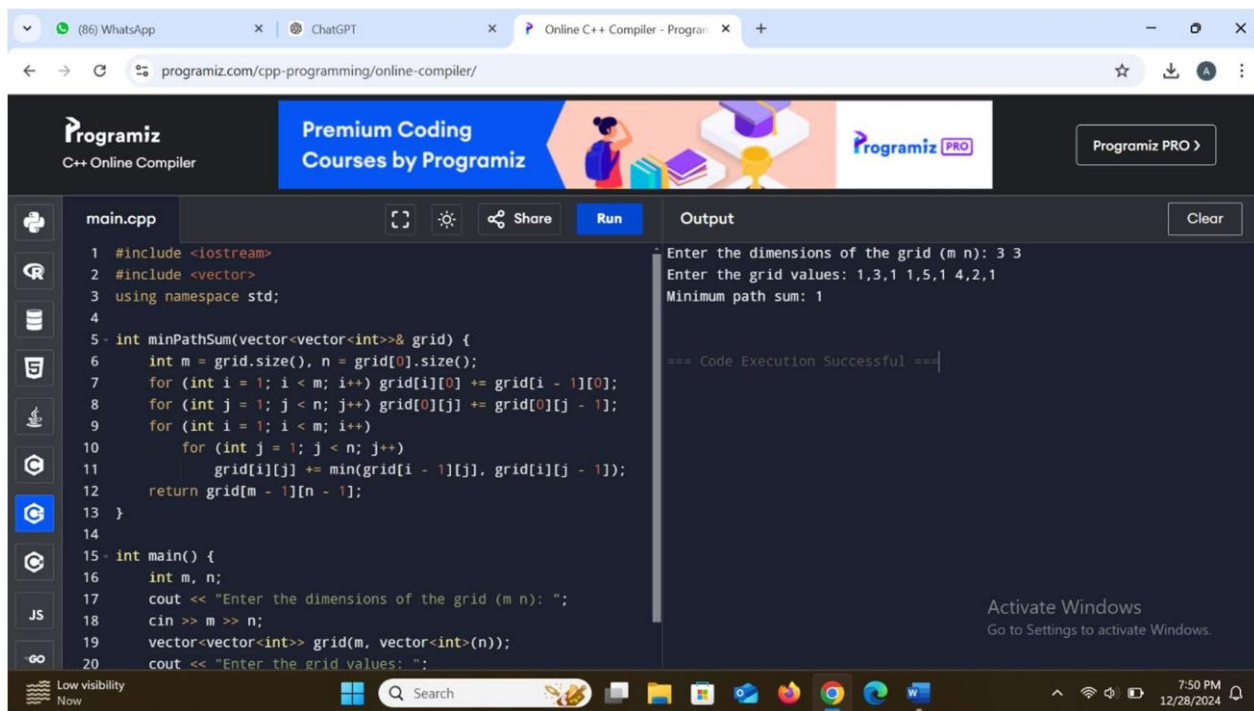
```
#include <iostream>
#include <vector>
using namespace std;
```

```
int minPathSum(vector<vector<int>>& grid) {
    int m = grid.size(), n = grid[0].size();
    for (int i = 1; i < m; i++) grid[i][0] += grid[i - 1][0];
    for (int j = 1; j < n; j++) grid[0][j] += grid[0][j - 1];
    for (int i = 1; i < m; i++)
        for (int j = 1; j < n; j++)
            grid[i][j] += min(grid[i - 1][j], grid[i][j - 1]);
    return grid[m - 1][n - 1];
}
```

```
int main() {
    int m, n;
    cout << "Enter the dimensions of the grid (m n): ";
    cin >> m >> n;
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
vector<vector<int>> grid(m, vector<int>(n));
cout << "Enter the grid values: ";
for (int i = 0; i < m; i++)
    for (int j = 0; j < n; j++)
        cin >> grid[i][j];
cout << "Minimum path sum: " << minPathSum(grid) << endl;
return 0;
}
```



The screenshot shows a web browser window with the URL `programiz.com/cpp-programming/online-compiler/`. The page features the Programiz logo and a navigation bar with options like 'Premium Coding Courses by Programiz' and 'Programiz PRO'. The main area is divided into a code editor on the left and an output window on the right. The code editor contains a C++ program for finding the minimum path sum in a grid. The output window shows the program's execution, including user input for grid dimensions (3x3) and values, and the resulting minimum path sum (1). The Windows taskbar at the bottom indicates the system time as 7:50 PM on 12/28/2024.

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int minPathSum(vector<vector<int>>& grid) {
6     int m = grid.size(), n = grid[0].size();
7     for (int i = 1; i < m; i++) grid[i][0] += grid[i - 1][0];
8     for (int j = 1; j < n; j++) grid[0][j] += grid[0][j - 1];
9     for (int i = 1; i < m; i++)
10         for (int j = 1; j < n; j++)
11             grid[i][j] += min(grid[i - 1][j], grid[i][j - 1]);
12     return grid[m - 1][n - 1];
13 }
14
15 int main() {
16     int m, n;
17     cout << "Enter the dimensions of the grid (m n): ";
18     cin >> m >> n;
19     vector<vector<int>> grid(m, vector<int>(n));
20     cout << "Enter the grid values: ";
```

Output:

```
Enter the dimensions of the grid (m n): 3 3
Enter the grid values: 1,3,1 1,5,1 4,2,1
Minimum path sum: 1

=== Code Execution Successful ===
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

7) Maximal Rectangle

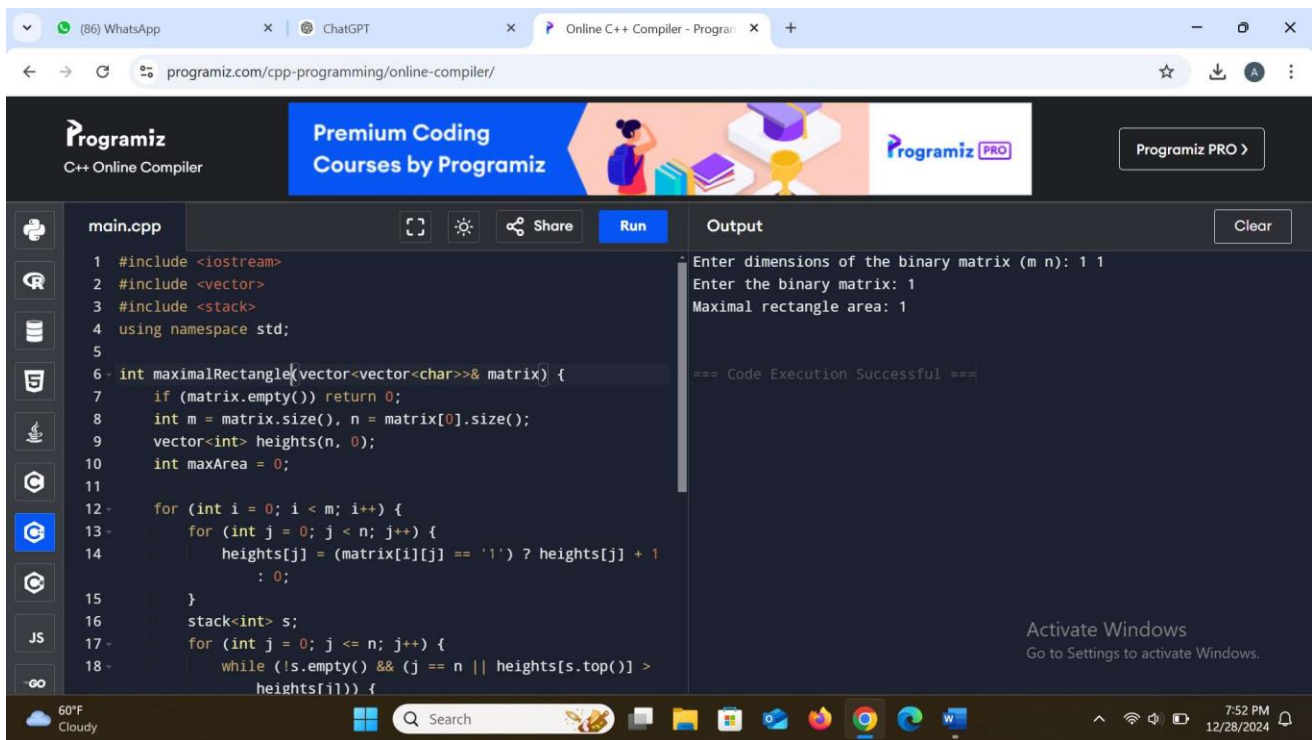
```
#include <iostream>
#include <vector>
#include <stack>
using namespace std;

int maximalRectangle(vector<vector<char>>& matrix) {
    if (matrix.empty()) return 0;
    int m = matrix.size(), n = matrix[0].size();
    vector<int> heights(n, 0);
    int maxArea = 0;

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            heights[j] = (matrix[i][j] == '1') ? heights[j] + 1 : 0;
        }
        stack<int> s;
        for (int j = 0; j <= n; j++) {
            while (!s.empty() && (j == n || heights[s.top()] > heights[j])) {
                int h = heights[s.top()];
                s.pop();
                int width = s.empty() ? j : j - s.top() - 1;
                maxArea = max(maxArea, h * width);
            }
            s.push(j);
        }
    }
    return maxArea;
}

int main() {
    int m, n;
    cout << "Enter dimensions of the binary matrix (m n): ";
    cin >> m >> n;
    vector<vector<char>> matrix(m, vector<char>(n));
    cout << "Enter the binary matrix: ";
    for (int i = 0; i < m; i++)
        for (int j = 0; j < n; j++)
            cin >> matrix[i][j];
    cout << "Maximal rectangle area: " << maximalRectangle(matrix) << endl;
    return 0;
}
```


DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



8) Dungeon Game

```
#include <iostream>
#include <vector>
#include <climits> // For INT_MAX
using namespace std;

int calculateMinimumHP(vector<vector<int>>& dungeon) {
    int m = dungeon.size(), n = dungeon[0].size();
    vector<vector<int>> dp(m + 1, vector<int>(n + 1, INT_MAX));
    dp[m][n - 1] = dp[m - 1][n] = 1;

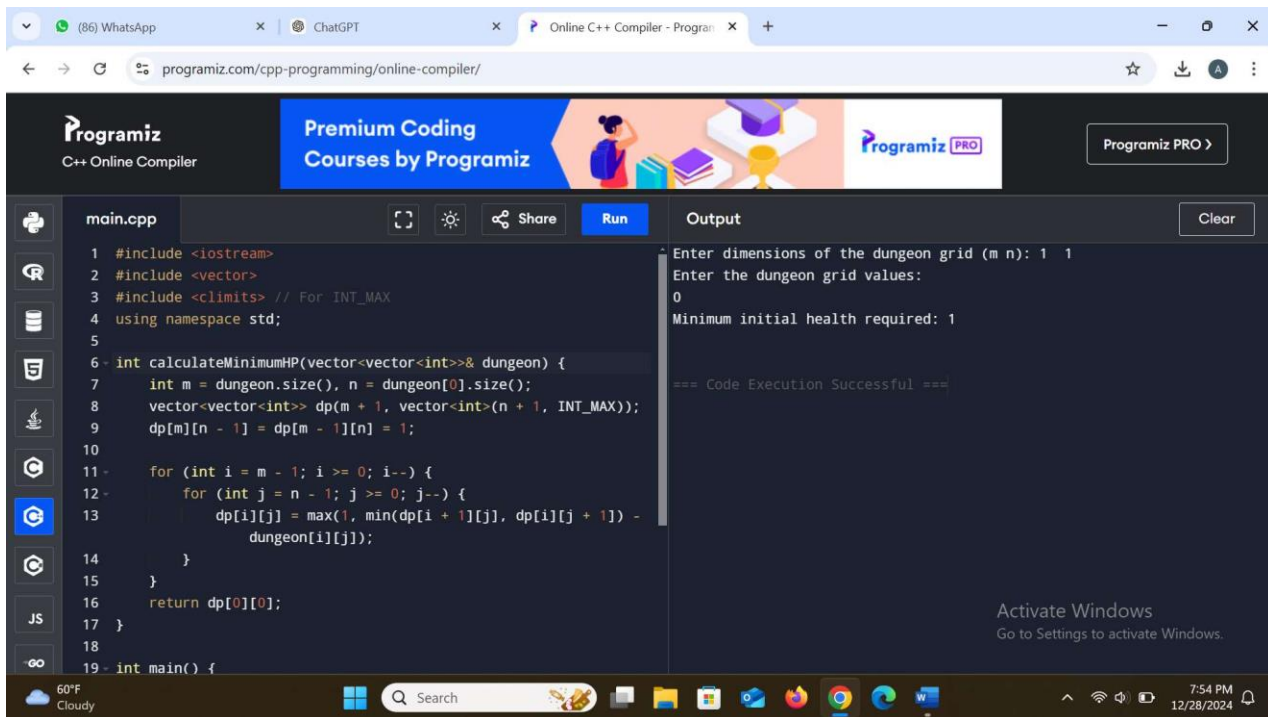
    for (int i = m - 1; i >= 0; i--) {
        for (int j = n - 1; j >= 0; j--) {
            dp[i][j] = max(1, min(dp[i + 1][j], dp[i][j + 1]) - dungeon[i][j]);
        }
    }
    return dp[0][0];
}

int main() {
    int m, n;
    cout << "Enter dimensions of the dungeon grid (m n): ";
    cin >> m >> n;
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
vector<vector<int>> dungeon(m, vector<int>(n));
cout << "Enter the dungeon grid values: " << endl;
for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
        cin >> dungeon[i][j];
    }
}

cout << "Minimum initial health required: " << calculateMinimumHP(dungeon) <<
endl;
return 0;
}
```



The screenshot displays the Programiz online C++ compiler interface. The browser address bar shows the URL `programiz.com/cpp-programming/online-compiler/`. The interface includes a header with the Programiz logo, navigation links for "Premium Coding Courses by Programiz" and "Programiz PRO", and a "Run" button. The main editor area shows a C++ program named `main.cpp` with the following code:

```
1 #include <iostream>
2 #include <vector>
3 #include <climits> // For INT_MAX
4 using namespace std;
5
6 int calculateMinimumHP(vector<vector<int>>& dungeon) {
7     int m = dungeon.size(), n = dungeon[0].size();
8     vector<vector<int>> dp(m + 1, vector<int>(n + 1, INT_MAX));
9     dp[m][n - 1] = dp[m - 1][n] = 1;
10
11     for (int i = m - 1; i >= 0; i--) {
12         for (int j = n - 1; j >= 0; j--) {
13             dp[i][j] = max(1, min(dp[i + 1][j], dp[i][j + 1]) -
14                             dungeon[i][j]);
15         }
16     }
17     return dp[0][0];
18 }
19 int main() {
```

The output window on the right shows the program's execution results:

```
Enter dimensions of the dungeon grid (m n): 1 1
Enter the dungeon grid values:
0
Minimum initial health required: 1

=== Code Execution Successful ===
```

The bottom of the screenshot shows a Windows taskbar with the date and time as 7:54 PM on 12/28/2024.

9) Sliding Puzzle

```
#include <iostream>
#include <vector>
#include <queue>
#include <unordered_set>
#include <string>
using namespace std;

int slidingPuzzle(vector<vector<int>>& board) {
    string target = "123450";
    string start = "";
    for (auto& row : board)
        for (int cell : row)
            start += to_string(cell);

    vector<vector<int>> dirs = {{1, 3}, {0, 2, 4}, {1, 5}, {0, 4}, {1, 3, 5}, {2, 4}};
    unordered_set<string> visited;
    queue<pair<string, int>> q;
    q.push({start, 0});
    visited.insert(start);

    while (!q.empty()) {
        auto [curr, steps] = q.front();
        q.pop();
        if (curr == target) return steps;

        int zeroPos = curr.find('0');
        for (int dir : dirs[zeroPos]) {
            string next = curr;
            swap(next[zeroPos], next[dir]);
            if (visited.find(next) == visited.end()) {
                visited.insert(next);
                q.push({next, steps + 1});
            }
        }
    }
    return -1; // Impossible to solve
}
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
int main() {  
    vector<vector<int>> board(2, vector<int>(3));  
    cout << "Enter the 2x3 board configuration row-wise (use 0 for the empty tile):" <<  
endl;  
    for (int i = 0; i < 2; i++)  
        for (int j = 0; j < 3; j++)  
            cin >> board[i][j];  
  
    int result = slidingPuzzle(board);  
    if (result == -1) {  
        cout << "The puzzle is unsolvable." << endl;  
    } else {  
        cout << "Minimum number of moves to solve the puzzle: " << result << endl;  
    }  
    return 0;  
}
```

The screenshot displays the Programiz online C++ compiler interface. The browser tabs at the top include (86) WhatsApp, ChatGPT, and Online C++ Compiler - Programiz. The address bar shows the URL `programiz.com/cpp-programming/online-compiler/`. The main interface features a dark theme with a sidebar on the left containing icons for file management and a search bar. The central editor area shows the code for `main.cpp`, which includes headers for `<iostream>`, `<vector>`, `<queue>`, `<unordered_set>`, and `<string>`. The code defines a `slidingPuzzle` function that takes a `vector<vector<int>>& board` and returns the minimum number of moves to solve the puzzle. The output window on the right shows the program's execution: it prompts for the 2x3 board configuration, receives the input `1 2 3` and `4 0 5`, and outputs `Minimum number of moves to solve the puzzle: 1`. A status message at the bottom of the output window indicates `=== Code Execution Successful ===`. The Windows taskbar at the bottom shows the system clock as 7:57 PM on 12/28/2024.

```
1 #include <iostream>  
2 #include <vector>  
3 #include <queue>  
4 #include <unordered_set>  
5 #include <string>  
6 using namespace std;  
7  
8 int slidingPuzzle(vector<vector<int>>& board) {  
9     string target = "123450";  
10    string start = "";  
11    for (auto& row : board)  
12        for (int cell : row)  
13            start += to_string(cell);  
14  
15    vector<vector<int>> dirs = {{1, 3}, {0, 2, 4}, {1, 5}, {0,  
16        4}, {1, 3, 5}, {2, 4}};  
17    unordered_set<string> visited;  
18    queue<pair<string, int>> q;  
19    q.push({start, 0});  
20    visited.insert(start);
```

Enter the 2x3 board configuration row-wise (use 0 for the empty tile):
1 2 3
4 0 5
Minimum number of moves to solve the puzzle: 1
=== Code Execution Successful ===

DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

10) Super Egg Drop

```
#include <iostream>
#include <vector>
using namespace std;

int superEggDrop(int k, int n) {
    vector<vector<int>> dp(k + 1, vector<int>(n + 1, 0));
    int moves = 0;

    while (dp[k][moves] < n) {
        moves++;
        for (int i = 1; i <= k; i++) {
            dp[i][moves] = dp[i - 1][moves - 1] + dp[i][moves - 1] + 1;
        }
    }
    return moves;
}

int main() {
    int k, n;
    cout << "Enter the number of eggs: ";
    cin >> k;
    cout << "Enter the number of floors: ";
    cin >> n;

    cout << "Minimum number of moves to find the critical floor: " <<
    superEggDrop(k, n) << endl;
    return 0;
}
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Browser tabs: (86) WhatsApp, ChatGPT, Online C++ Compiler - Program

Address bar: programiz.com/cpp-programming/online-compiler/

Programiz C++ Online Compiler

Premium Coding Courses by Programiz

Programiz PRO

main.cpp

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int superEggDrop(int k, int n) {
6     vector<vector<int>> dp(k + 1, vector<int>(n + 1, 0));
7     int moves = 0;
8
9     while (dp[k][moves] < n) {
10         moves++;
11         for (int i = 1; i <= k; i++) {
12             dp[i][moves] = dp[i - 1][moves - 1] + dp[i][moves - 1] + 1;
13         }
14     }
15     return moves;
16 }
17
18 int main() {
19     int k, n;
```

Run

Output

Enter the number of eggs: 1
Enter the number of floors: 2
Minimum number of moves to find the critical floor: 2

=== Code Execution Successful ===

Activate Windows
Go to Settings to activate Windows.

60°F Cloudy Search 7:58 PM 12/28/2024