```cpp
/**
 * @file main.cpp
 * @author Christian Prather
 * @brief Testing algorithm for the optimization algorithm
 * @version 0.1
 * @date 2020-11-12
 *
 * @copyright Copyright (c) 2020
 *
 */
#include <iostream>
using namespace std;
/// Enum defines
#define FORWARD 0
#define RIGHT 1
#define LEFT 2
#define DISTANCE_SEG 10

int movesCount = 6;
// Global array for tracking move order (move, distance) or (move, degree)
int moveList[50] = {FORWARD, FORWARD, RIGHT, RIGHT, FORWARD, FORWARD};

int optimizedMoves[50];

void optimize()
{
    /// Key patterns 0 = F, 1 = R, 2 = L, 3 = DELETE
    int keyPatterns_6[2][6] = {{0, 0, 1, 1, 0, 0}, {2, 0, 1, 1, 0, 2}};
    int keyPatterns_5[2][5] = {{2, 0, 1, 1, 0}, {0, 1, 1, 0, 2}};
    int keyPatterns_4[1][4] = {{0, 1, 1, 0}};

    int optimizedPattern_6[1][8] = {{FORWARD, 2 * DISTANCE_SEG, RIGHT, 90, RIGHT, 90, FORWARD, DISTANCE_SEG}};
    int optimizedPattern_5[2][2] = {{RIGHT, 90}, {RIGHT, 90}};
    int optimizedPatter_4[1][4] = {{LEFT, 90, LEFT, 90}};
    /** This is going to be checking in a priority tree fashion given highest priority
    * given highest priority patterns are 6 long then 5 long then 4 I can batch this
    */

    for (int i = 0; i < movesCount; i++)
    {
        /// Get next move in explored list
        // int move = moveList[i];
        /// Get next 6 moves if enough in list

        // Check 6 out first
        int future[6];
        for (int j = 0; j < 6; j++)
        {
            if ((j + i) < movesCount)
            {
                /// j (0-5) i (0-movesCount)
                future[j] = moveList[j + i];
                cout << "Move: " << future[j] << endl;
            }
        }
        int tracker = 0;
        for (auto potential : keyPatterns_6)
        {
            bool match = true;
            for (int m = 0; m < 6; m++)
            {
                if (future[m] != potential[m])
                {
                    match = false;
                }
            }
            if (match)
            {
                cout << "Matched " << tracker << endl;
                int keyPatternLength = (sizeof(potential) / sizeof(potential[0]));
                // Insert optimized move
                for (int x = 0; x < (sizeof(optimizedPattern_6[tracker]) / sizeof(optimizedPattern_6[tracker][0])); x++)
                {
                    if (optimizedPattern_6[tracker][x] != 3)
                    {
                        optimizedMoves[x] = optimizedPattern_6[tracker][x];
                    }
                }
                i = i+ 6;
                break;
            }
            tracker = tracker + 1;
        }

 ///////////////////////////////////////////////////////////////////////////////
        // Check 5 out first
        int future_5[5];
        for (int j = 0; j < 5; j++)
        {
            if ((j + i) < movesCount)
            {
                /// j (0-5) i (0-movesCount)
                future_5[j] = moveList[j + i];
                cout << "Move5: " << future_5[j] << endl;
            }
        }
        tracker = 0;
        for (auto potential : keyPatterns_6)
        {
            bool match = true;
            for (int m = 0; m < 5; m++)
            {
                if (future_5[m] != potential[m])
                {
                    match = false;
                }
            }
            if (match)
            {
                cout << "Matched " << tracker << endl;
                int keyPatternLength = (sizeof(potential) / sizeof(potential[0]));
                // Insert optimized move
                for (int x = 0; x < (sizeof(optimizedPattern_6[tracker]) / sizeof(optimizedPattern_6[tracker][0])); x++)
                {
                    if (optimizedPattern_6[tracker][x] != 3)
                    {
                        optimizedMoves[x] = optimizedPattern_6[tracker][x];
                    }
                }
                i = i+ 5;
                break;
            }
            tracker = tracker + 1;
        }

 ////////////////////////////////////////////////////////////////////////////////
        // Check 4 out first
        int future_4[4];
        for (int j = 0; j < 4; j++)
        {
            if ((j + i) < movesCount)
            {
                /// j (0-5) i (0-movesCount)
                future_4[j] = moveList[j + i];
                cout << "Move4: " << future_4[j] << endl;
            }
        }
        tracker = 0;
        for (auto potential : keyPatterns_6)
        {
            bool match = true;
            for (int m = 0; m < 4; m++)
            {
                if (future_4[m] != potential[m])
                {
                    match = false;
                }
            }
            if (match)
            {
                cout << "Matched " << tracker << endl;
                int keyPatternLength = (sizeof(potential) / sizeof(potential[0]));
                // Insert optimized move
                for (int x = 0; x < (sizeof(optimizedPattern_6[tracker]) / sizeof(optimizedPattern_6[tracker][0])); x++)
                {
                    if (optimizedPattern_6[tracker][x] != 3)
                    {
                        optimizedMoves[x] = optimizedPattern_6[tracker][x];
                    }
                }
                i = i+4;
                break;
            }
            tracker = tracker + 1;
        }
    }
}
```

```cpp

int main()
{
    optimize();
    cout << "Optimized" << endl;
    for (auto move : optimizedMoves)
    {
        cout << move << ", ";
    }
}
```