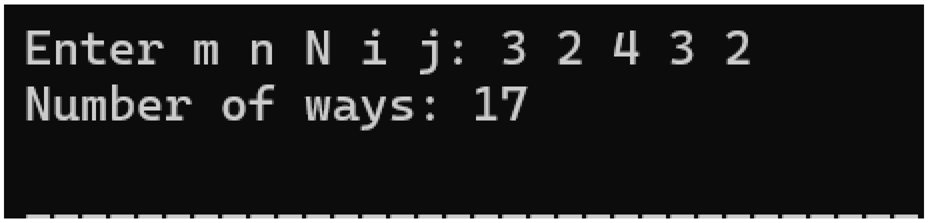
Program-11

```
#include <stdio.h>
#define MOD 1000000007
int findPaths(int m, int n, int N, int i, int j) {
    int dp[N + 1][m][n];
    for (int k = 0; k <= N; k++)
        for (int r = 0; r < m; r++)
            for (int c = 0; c < n; c++)
                dp[k][r][c] = 0;
    dp[0][i][j] = 1;
    int result = 0;
    int dir[4][2] = {{1,0},{-1,0},{0,1},{0,-1}};
    for (int step = 1; step <= N; step++) {
        for (int r = 0; r < m; r++) {
            for (int c = 0; c < n; c++) {
                if (dp[step-1][r][c] > 0) {
                    for (int d = 0; d < 4; d++) {
                        int nr = r + dir[d][0], nc = c + dir[d][1];
                        if (nr < 0 || nr >= m || nc < 0 || nc >= n)
                            result = (result + dp[step-1][r][c]) % MOD;
                        else
                            dp[step][nr][nc] = (dp[step][nr][nc] + dp[step-1][r][c]) % MOD;
                    }
                }
            }
        }
    }
    return result;
}
int main() {
    int m, n, N, i, j;
    printf("Enter m n N i j: ");
    scanf("%d %d %d %d %d", &m, &n, &N, &i, &j);
    printf("Number of ways: %d\n", findPaths(m, n, N, i, j));
    return 0;
}
```

Output:

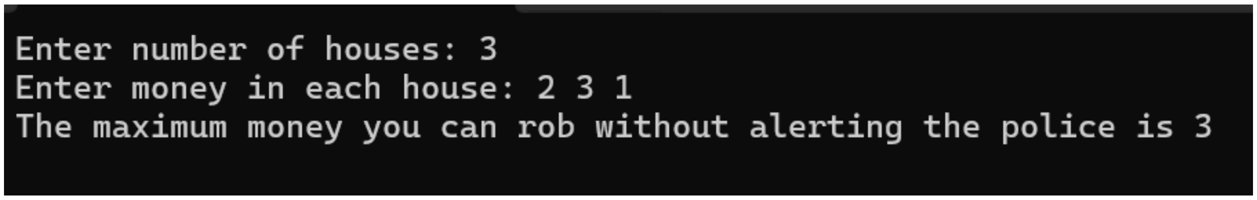


```
Enter m n N i j: 3 2 4 3 2
Number of ways: 17
```

Program-12:

```
#include <stdio.h>
int max(int a, int b) {
    return (a > b) ? a : b;
}
int robLinear(int nums[], int start, int end) {
    int prev1 = 0, prev2 = 0;
    for (int i = start; i <= end; i++) {
        int temp = prev1;
        prev1 = max(prev1, prev2 + nums[i]);
        prev2 = temp;
    }
    return prev1;
}
int rob(int nums[], int n) {
    if (n == 0) return 0;
    if (n == 1) return nums[0];
    return max(robLinear(nums, 0, n - 2), robLinear(nums, 1, n - 1));
}
int main() {
    int n;
    printf("Enter number of houses: ");
    scanf("%d", &n);
    int nums[n];
    printf("Enter money in each house: ");
    for (int i = 0; i < n; i++)
        scanf("%d", &nums[i]);
    int result = rob(nums, n);
    printf("The maximum money you can rob without alerting the police is %d\n", result);
    return 0;
}
```

Output:

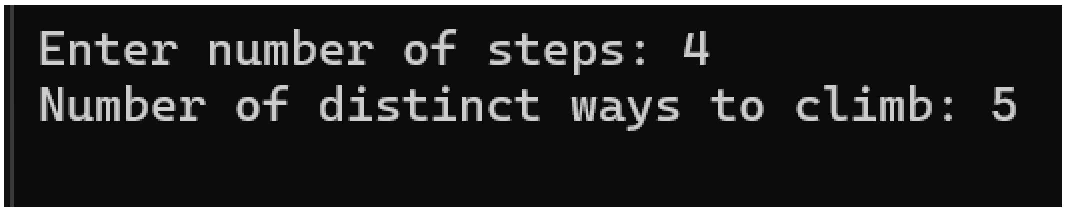


```
Enter number of houses: 3
Enter money in each house: 2 3 1
The maximum money you can rob without alerting the police is 3
```

Program-13:

```c
#include <stdio.h>
int climbStairs(int n) {
    if (n <= 2) return n;
    int a = 1, b = 2, c;
    for (int i = 3; i <= n; i++) {
        c = a + b;
        a = b;
        b = c;
    }
    return b;
}

int main() {
    int n;
    printf("Enter number of steps: ");
    scanf("%d", &n);
    printf("Number of distinct ways to climb: %d\n", climbStairs(n));
    return 0;
}
```

Output:

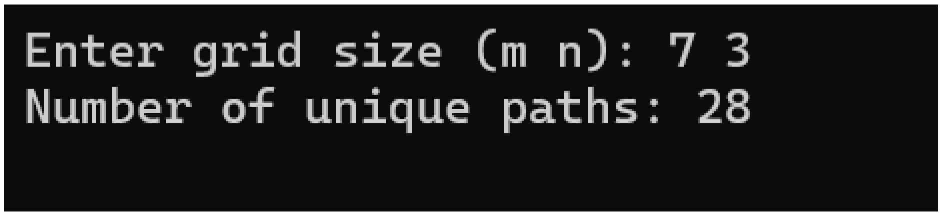


```
Enter number of steps: 4
Number of distinct ways to climb: 5
```

Program 14:

```
#include <stdio.h>
int uniquePaths(int m, int n) {
    int dp[m][n];
    for (int i = 0; i < m; i++)
        dp[i][0] = 1;
    for (int j = 0; j < n; j++)
        dp[0][j] = 1;
    for (int i = 1; i < m; i++)
        for (int j = 1; j < n; j++)
            dp[i][j] = dp[i-1][j] + dp[i][j-1];

    return dp[m-1][n-1];
}
int main() {
    int m, n;
    printf("Enter grid size (m n): ");
    scanf("%d %d", &m, &n);
    printf("Number of unique paths: %d\n", uniquePaths(m, n));
    return 0;
}
```

Output:

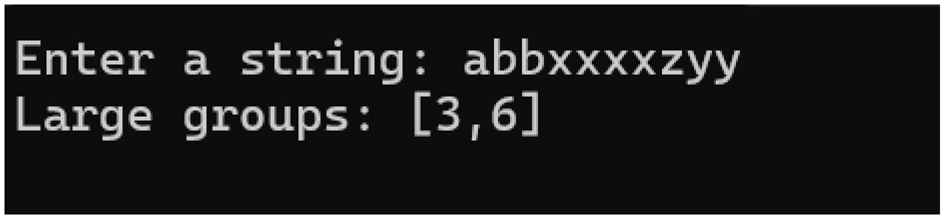


```
Enter grid size (m n): 7 3
Number of unique paths: 28
```

Program 15:

```
#include <stdio.h>
#include <string.h>
int main() {
    char s[100];
    printf("Enter a string: ");
    scanf("%s", s);
    int n = strlen(s);
    int start = 0;

    printf("Large groups: ");
    for (int i = 1; i <= n; i++) {
        if (s[i] != s[i - 1]) {
            if (i - start >= 3)
                printf("[%d,%d] ", start, i - 1);
            start = i;
        }
    }
    printf("\n");
    return 0;
}
```

Output:

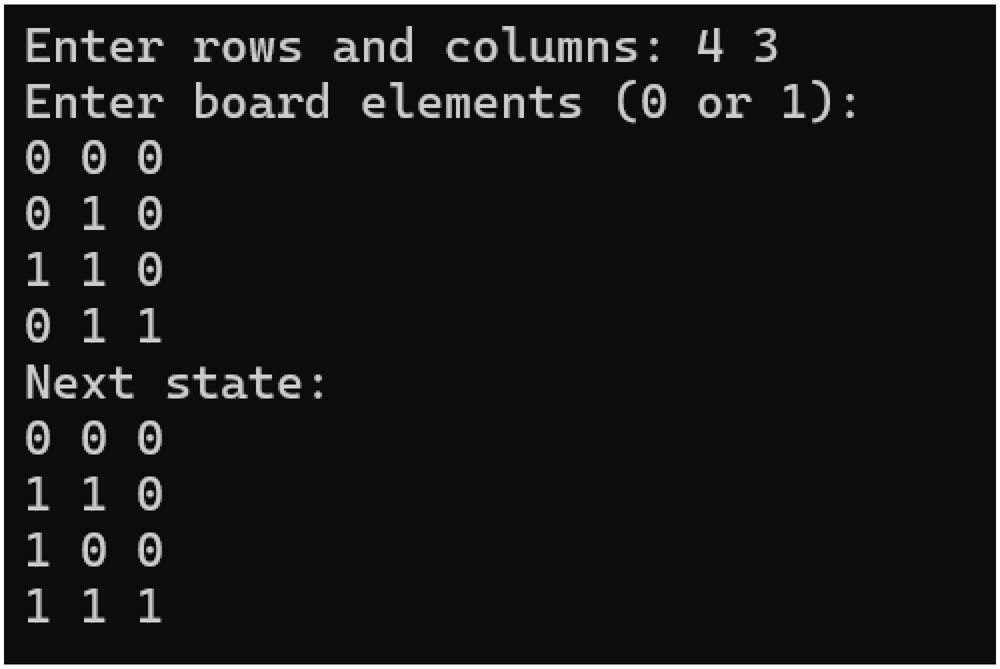


```
Enter a string: abbxxxxzyy
Large groups: [3,6]
```

Program 16:

```
#include <stdio.h>
int main() {
    int m, n;
    printf("Enter rows and columns: ");
    scanf("%d %d", &m, &n);
    int board[m][n], next[m][n];
    printf("Enter board elements (0 or 1):\n");
    for (int i = 0; i < m; i++)
        for (int j = 0; j < n; j++)
            scanf("%d", &board[i][j]);
    int dx[] = {-1,-1,-1,0,0,1,1,1};
    int dy[] = {-1,0,1,-1,1,-1,0,1};
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            int live = 0;
            for (int k = 0; k < 8; k++) {
                int ni = i + dx[k], nj = j + dy[k];
                if (ni >= 0 && ni < m && nj >= 0 && nj < n)
                    live += board[ni][nj];
            }
            if (board[i][j] == 1) {
                next[i][j] = (live == 2 || live == 3) ? 1 : 0;
            } else {
                next[i][j] = (live == 3) ? 1 : 0;
            }
        }
    }
    printf("Next state:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++)
            printf("%d ", next[i][j]);
        printf("\n");
    }
    return 0;
}
```

Output:

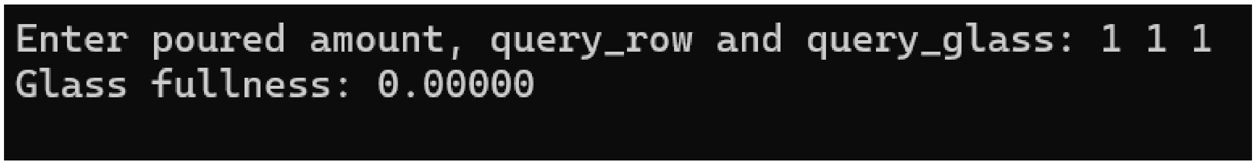


```
Enter rows and columns: 4 3
Enter board elements (0 or 1):
0 0 0
0 1 0
1 1 0
0 1 1
Next state:
0 0 0
1 1 0
1 0 0
1 1 1
```

Program 17:

```
#include <stdio.h>

int main() {

    int poured, query_row, query_glass;

    printf("Enter poured amount, query_row and query_glass: ");

    scanf("%d %d %d", &poured, &query_row, &query_glass);

    double tower[101][101] = {0.0};

    tower[0][0] = poured;

    for (int i = 0; i < 100; i++) {

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

            if (tower[i][j] > 1.0) {

                double excess = (tower[i][j] - 1.0) / 2.0;

                tower[i+1][j] += excess;

                tower[i+1][j+1] += excess;

                tower[i][j] = 1.0;

            }

        }

    }

    double result = tower[query_row][query_glass];

    if (result > 1.0) result = 1.0;

    printf("Glass fullness: %.5f\n", result);

    return 0;

}
```

Output:



```
Enter poured amount, query_row and query_glass: 1 1 1
Glass fullness: 0.00000
```