

```
main.cpp
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int knapSackRec (int W, int wt[], int val[], int index, int **dp)
5 {
6     if (index < 0)
7         return 0;
8     if (dp[index][W] != -1)
9         return dp[index][W];
10
11     if (wt[index] > W){
12         dp[index][W] = knapSackRec (W, wt, val, index - 1, dp);
13         return dp[index][W];
14     }
15     else
16     {
17         dp[index][W] = max (val[index]
18                             + knapSackRec (W - wt[index], wt, val,
19                                             index - 1, dp),
20                             knapSackRec (W, wt, val, index - 1, dp));
21         return dp[index][W];
22     }
23 }
24
25 int knapSack (int W, int wt[], int val[], int n)
26 {
27     int **dp;
28     dp = new int *[n];
29
30     for (int i = 0; i < n; i++)
31         dp[i] = new int[W + 1];
32
33     for (int i = 0; i < n; i++)
34         for (int j = 0; j < W + 1; j++)
35             dp[i][j] = -1;
36     return knapSackRec (W, wt, val, n - 1, dp);
37 }
38
39 int main ()
40 {
41     int profit[] = { 60, 100, 120 };
42     int weight[] = { 10, 20, 30 };
43     int W = 50;
44     int n = sizeof (profit) / sizeof (profit[0]);
45     cout << knapSack (W, weight, profit, n);
46     return 0;
47 }
```

220

...Program finished with exit code 0  
Press ENTER to exit console.

```
main.cpp
1 #include <bits/stdc++.h>
2 #define N 4
3 using namespace std;
4
5 void printSolution(int board[N][N])
6 {
7     for (int i = 0; i < N; i++) {
8         for (int j = 0; j < N; j++)
9             if (board[i][j])
10                 cout << "Q ";
11         else cout << ". ";
12         printf("\n");
13     }
14 }
15
16 bool isSafe(int board[N][N], int row, int col)
17 {
18     int i, j;
19
20     for (i = 0; i < col; i++)
21         if (board[row][i])
22             return false;
23
24     for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
25         if (board[i][j])
26             return false;
27
28     for (i = row, j = col; j >= 0 && i < N; i++, j--)
29         if (board[i][j])
30             return false;
31
32     return true;
33 }
34
35 bool solveNQUtil(int board[N][N], int col)
36 {
37     if (col >= N)
38         return true;
39
40     for (int i = 0; i < N; i++) {
41         if (isSafe(board, i, col)) {
42             board[i][col] = 1;
43             if (solveNQUtil(board, col + 1))
44                 return true;
45             board[i][col] = 0;
46         }
47     }
48     return false;
49 }
50 bool solveNQ()
51 {
52     int board[N][N] = { { 0, 0, 0, 0 },
53                          { 0, 0, 0, 0 },
54                          { 0, 0, 0, 0 },
55                          { 0, 0, 0, 0 } };
56
57     if (solveNQUtil(board, 0) == false) {
58         cout << "Solution does not exist";
59         return false;
60     }
61
62     printSolution(board);
63     return true;
64 }
65
66 int main()
67 {
68     solveNQ();
69     return 0;
70 }
71
72
```

```
. . Q .
Q . . .
. . . Q
. Q . .
```

...Program finished with exit code 0  
Press ENTER to exit console.

```
main.cpp
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define MAX 4
4 #define MAX_THREAD 4
5
6 int matA[MAX][MAX];
7 int matB[MAX][MAX];
8 int matC[MAX][MAX];
9 int step_i = 0;
10
11 void* multi(void* arg)
12 {
13     int i = step_i++;
14
15     for (int j = 0; j < MAX; j++)
16         for (int k = 0; k < MAX; k++)
17             matC[i][j] += matA[i][k] * matB[k][j];
18 }
19
20 int main()
21 {
22     for (int i = 0; i < MAX; i++) {
23         for (int j = 0; j < MAX; j++) {
24             matA[i][j] = rand() % 10;
25             matB[i][j] = rand() % 10;
26         }
27     }
28
29     cout << endl;
30     << "Matrix A" << endl;
31     for (int i = 0; i < MAX; i++) {
32         for (int j = 0; j < MAX; j++)
33             cout << matA[i][j] << " ";
34         cout << endl;
35     }
36
37     cout << endl;
38     << "Matrix B" << endl;
39     for (int i = 0; i < MAX; i++) {
40         for (int j = 0; j < MAX; j++)
41             cout << matB[i][j] << " ";
42         cout << endl;
43     }
44
45     pthread_t threads[MAX_THREAD];
46
47     for (int i = 0; i < MAX_THREAD; i++) {
48         int* p;
49         pthread_create(&threads[i], NULL, multi, (void*)(p));
50     }
51     for (int i = 0; i < MAX_THREAD; i++)
52         pthread_join(threads[i], NULL);
53
54     cout << endl;
55     << "Multiplication of A and B" << endl;
56     for (int i = 0; i < MAX; i++) {
57         for (int j = 0; j < MAX; j++)
58             cout << matC[i][j] << " ";
59         cout << endl;
60     }
61     return 0;
62 }
63
```

```
Matrix A
3 7 3 6
9 2 0 3
0 2 1 7
2 2 7 9

Matrix B
6 5 5 2
1 7 9 6
6 6 8 9
0 3 5 2

Multiplication of A and B
43 100 132 87
56 68 78 36
8 41 61 35
56 93 129 97

...Program finished with exit code 0
Press ENTER to exit console.
```