Run | Debug | Stop | Share | Save | { } Beautify

main.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

int knapSackRec (int W, int wt[], int val[], int index, int **dp)
{
  if (index < 0)
    return 0;
  if (dp[index][W] != -1)
    return dp[index][W];

  if (wt[index] > W){
      dp[index][W] = knapSackRec (W, wt, val, index - 1, dp);
      return dp[index][W];
    }
  else
    {
      dp[index][W] = max (val[index]
              + knapSackRec (W - wt[index], wt, val,
                    index - 1, dp),
              knapSackRec (W, wt, val, index - 1, dp));
      return dp[index][W];
    }
}

int knapSack (int W, int wt[], int val[], int n)
{
  int **dp;
  dp = new int *[n];

  for (int i = 0; i < n; i++)
    dp[i] = new int[W + 1];

  for (int i = 0; i < n; i++)
    for (int j = 0; j < W + 1; j++)
      dp[i][j] = -1;
  return knapSackRec (W, wt, val, n - 1, dp);
}

int main ()
{
  int profit[] = { 60, 100, 120 };
  int weight[] = { 10, 20, 30 };
  int W = 50;
  int n = sizeof (profit) / sizeof (profit[0]);
  cout << knapSack (W, weight, profit, n);
  return 0;
}
```

```
220

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

Run Debug Stop Share Save { } Beautify

main.cpp

```cpp
#include <bits/stdc++.h>
#define N 4
using namespace std;

void printSolution(int board[N][N])
{
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++)
        if(board[i][j])
            cout << "Q ";
        else cout<<". ";
        printf("\n");
    }
}

bool isSafe(int board[N][N], int row, int col)
{
    int i, j;

    for (i = 0; i < col; i++)
        if (board[row][i])
            return false;

    for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
        if (board[i][j])
            return false;

    for (i = row, j = col; j >= 0 && i < N; i++, j--)
        if (board[i][j])
            return false;

    return true;
}

bool solveNQUtil(int board[N][N], int col)
{
    if (col >= N)
        return true;

    for (int i = 0; i < N; i++) {
        if (isSafe(board, i, col)) {
            board[i][col] = 1;
            if (solveNQUtil(board, col + 1))
                return true;
            board[i][col] = 0;
        }
    }
    return false;
}
bool solveNQ()
{
    int board[N][N] = { { 0, 0, 0, 0 },
                        { 0, 0, 0, 0 },
                        { 0, 0, 0, 0 },
                        { 0, 0, 0, 0 } };

    if (solveNQUtil(board, 0) == false) {
        cout << "Solution does not exist";
        return false;
    }

    printSolution(board);
    return true;
}

int main()
{
    solveNQ();
    return 0;
}
```

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


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

Run | Debug | Stop | Share | Save | {} Beautify

main.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
#define MAX 4
#define MAX_THREAD 4

int matA[MAX][MAX];
int matB[MAX][MAX];
int matC[MAX][MAX];
int step_i = 0;

void* multi(void* arg)
{
    int i = step_i++;

    for (int j = 0; j < MAX; j++)
    for (int k = 0; k < MAX; k++)
        matC[i][j] += matA[i][k] * matB[k][j];
}

int main()
{
    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < MAX; j++) {
            matA[i][j] = rand() % 10;
            matB[i][j] = rand() % 10;
        }
    }

    cout << endl
        << "Matrix A" << endl;
    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < MAX; j++)
            cout << matA[i][j] << " ";
        cout << endl;
    }

    cout << endl
        << "Matrix B" << endl;
    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < MAX; j++)
            cout << matB[i][j] << " ";
        cout << endl;
    }

    pthread_t threads[MAX_THREAD];

    for (int i = 0; i < MAX_THREAD; i++) {
        int* p;
        pthread_create(&threads[i], NULL, multi, (void*)(p));
    }
    for (int i = 0; i < MAX_THREAD; i++)
        pthread_join(threads[i], NULL);

    cout << endl
        << "Multiplication of A and B" << endl;
    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < MAX; j++)
            cout << matC[i][j] << " ";
        cout << endl;
    }
    return 0;
}
```

```
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.
```