maxBPM.cpp

```cpp
#include <iostream>
#include <string.h>
using namespace std;

// M is number of applicants and N is number of jobs
#define M 6
#define N 6

// A DFS based recursive function that returns true if a
// matching for vertex u is possible
bool bpm(bool bpGraph[M][N], int u, bool seen[], int matchR[]) {
    // Try every job one by one
    for (int v = 0; v < N; v++)  {
        // If applicant u is interested in job v and v is
        // not visited
        if (bpGraph[u][v] && !seen[v]) {
            seen[v] = true; // Mark v as visited

            if (matchR[v] < 0 || bpm(bpGraph, matchR[v], seen, matchR)) {
                matchR[v] = u;
                return true;
            }
        }
    }
    return false;
}

// Returns maximum number of matching from M to N
int maxBPM(bool bpGraph[M][N]) {

    int matchR[N];

    memset(matchR, -1, sizeof(matchR));

    int result = 0; // Count of jobs assigned to applicants
    for (int u = 0; u < M; u++) {
        bool seen[N];
        memset(seen, 0, sizeof(seen));

        // Find if the applicant 'u' can get a job
        if (bpm(bpGraph, u, seen, matchR))
            result++;
    }
    return result;
}

// Driver program to test above functions
int main() {
    // Let us create a bpGraph shown in the above example
    bool bpGraph[M][N] = {  {0, 1, 1, 0, 0, 0},
                            {1, 0, 0, 1, 0, 0},
                            {0, 0, 1, 0, 0, 0},
                            {0, 0, 1, 1, 0, 0},
                            {0, 0, 0, 0, 0, 0},
                            {0, 0, 0, 0, 0, 1}
    };

    cout << "Maximum number of applicants that can get job is " << maxBPM(bpGraph);

    return 0;
}
```