1)

```
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
int n,m;
int countt=0;
#define CN countt++
int p[100],w[100];


struct PW
{
    vector <pair<int,int>>S;
}X[100];

void display_pair(vector <pair<int,int>> x)
{
    vector <pair<int,int>> :: iterator it;
    for(it=x.begin();it!=x.end();++it)
    {
       cout<<"{"; cout<<(*it).first;
        cout<<",";
        cout<<(*it).second;cout<<"} ";
    }
    cout<<endl;
}

void purge()
{
    vector <pair<int,int>>:: iterator i;
    vector <pair<int,int>>:: iterator j;

    for(int k=1;k<=n;k++)
    {

          for(i=X[k].S.begin(); i!= X[k].S.end();++i)
          {
            CN;
            for(j=i+1; j!= X[k].S.end();++j)
            {
                CN;
                if((*i).first<=(*j).first && (*i).second >= (*j).second)
                {CN;X[k].S.erase(i--);j--;CN;}
            }
          }
    }
}


bool PairMatch(vector <pair<int,int>> z, int pp, int ww)
{
    vector <pair<int,int>> :: iterator zi;
    for(zi=z.begin(); zi!=z.end(); ++zi)
    {
        CN;
        if((*zi).first==pp && (*zi).second== ww)
       {CN; return true;CN;}
    }
    return false;
}
```

```
void traceBack()
{
        vector <pair<int,int>> :: iterator it;
          vector <pair<int,int>> :: iterator c;
        int solVect[n+1];
        it=max_element(X[n].S.begin(),X[n].S.end());CN; //Find the largest
pair
        int pp=(*it).first; int ww=(*it).second;CN;

        cout<<"MAX PROFIT: "<<pp<<endl;
        cout<<"TOTAL WEIGHT OCCUPIED: "<<ww<<endl;

         for(int p=0; p<n;p++)
         {
            solVect[p]=0;
         }

        //Start from the n-1 set
        for(int i=n-1;i>=0;i--)
        {
            CN;
          bool status=PairMatch(X[i].S,pp,ww);CN;//check if the set is
present or no

          if(status)
          {
            CN;
            solVect[i]=0;CN;
          }
          else
          {
            solVect[i]=1;CN;
            pp=pp - p[i];CN;
             ww=ww - w[i];CN;
          }

        }
        cout<<"SOLUTION VECTOR: {";
        for(int p=0; p<n;p++)
        {
            CN;
            cout<<solVect[p];CN;
            if(p!=n-1) cout<<",";
        }cout<<"}"<<endl;

}

void Dk()
{
    //S0 initially;
     vector <pair<int,int>> :: iterator it;
    X[0].S.push_back(make_pair(0,0));CN;

    int next=1; //next is S1...

    for(int i=0;i<n;i++)
    {
        CN;
        for(it=X[next-1].S.begin();it!=X[next-1].S.end();++it) //Merge
previous S1n-1
        {
            CN;
            X[next].S.push_back(make_pair((*it).first,(*it).second));CN;
```

```
        }

         for(it=X[next-1].S.begin();it!=X[next-1].S.end();++it)
//Calculate the remaining and merge
        {
            CN;
            if(((*it).second)+w[i] > m)
            {CN;continue;}

X[next].S.push_back(make_pair(((*it).first+p[i]),((*it).second)+w[i]));CN;
        };
        next++;CN;

    }

    purge();cout<<endl;CN;
    for(int k=0;k<=n;k++) //Display the subsets
    {
        CN;
          cout<<"S"<<k<<":   ";CN;
        display_pair(X[k].S);CN;
        cout<<endl;
    }
    traceBack();


}
int main()
{

    cout<<"ENTER NUMBER OF ELEMENTS : ";
    cin>>n;
    cout<<"ENTER PROFIT AND WEIGHT OF ELEMENTS: "<<endl;
    for(int i=0;i<n;i++)
    {
        cin>>p[i]>>w[i];
    }

    cout<<"ENTER THE BAG CAPACITY : ";
    cin>>m;

    Dk();

    cout<<"\nSTEP COUNT: "<<countt;
}
```

OUTPUT

```
ENTER NUMBER OF ELEMENTS : 4
ENTER PROFIT AND WEIGHT OF ELEMENTS:
10 1
12 2
15 2
20 3
ENTER THE BAG CAPACITY : 5

S0:   {0,0}

S1:   {0,0} {10,1}

S2:   {0,0} {10,1} {12,2} {22,3}

S3:   {0,0} {10,1} {15,2} {25,3} {27,4} {37,5}

S4:   {0,0} {10,1} {15,2} {25,3} {37,5} {20,3} {30,4} {35,5}

MAX PROFIT: 37
TOTAL WEIGHT OCCUPIED: 5
SOLUTION VECTOR: {1,1,1,0}

STEP COUNT: 267
PS C:\Users\Lloyd\Desktop\MADF\EXPT 11>
```

1)

```
#include<iostream>
#include<math.h>
#include<iomanip>
#define MAX 10
#define CNT cntt++
using namespace std;

int outcome[MAX],n,count=0,cntt=0;
int TupleDATABASE[1000][MAX],dataBcnt=1;

void printChessboard()
{
    int x;
    cout<<"ENTER INDEX NUMBER OF THE TUPLE TO BE PRINTED: ";
    cin>>x;cout<<endl;
    int i,j;
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            if(TupleDATABASE[x][i]==j)
            cout<<"Q"<<" ";
            else
            cout<<"^"<<" ";
        }
        cout<<endl;
    }
}

void Print_Outcome()
{

    cout<<setw(4)<<++count<<"| "<<"{";
    int i,j;
    for(i=1;i<=n;i++,j++)
    {
        TupleDATABASE[dataBcnt][i]=outcome[i];

        cout<<outcome[i];
        if(i!=n)
        cout<<",";
    }
    cout<<"}      ";

    dataBcnt++;
    if(count%2==0)
    cout<<endl;
}

bool Place(int k,int i)
{
    int j;
    for(j=1;j<=k-1;j++)
    {
        CNT;
        if(outcome[j]==i || abs(outcome[j]-i)==abs(j-k))
        {CNT;return false;}
    }
    return true;
}
```

```
void NQueens(int k,int n)
{
    int i;
    for(i=1;i<=n;i++)
    {
        CNT;
        if(Place(k,i))
        {
            CNT;
            outcome[k]=i;CNT;
            if(k==n)
            {CNT;Print_Outcome();}
            else
            {CNT;NQueens(k+1,n);}
        }
    }
}

void setPrint()
{
    int i;
    cout<<"  {";
    for(i=1;i<=n;i++)
    {
        cout<<"X"<<i;
        if(i!=n)
        cout<<",";
    }
    cout<<"} "<<endl;
}

int main()
{
    int choice;
    cout<<"ENTER VALUE OF N: ";
    cin>>n;
    cout<<endl<<"      POSSIBLE SOLUTIONS";
    setPrint();cout<<endl;
    NQueens(1,n);
      cout<<"STEP COUNT: "<<cntt<<endl;


    while(1)
    {

cout<<"\n\n_____________________________OPTIONS_____________________________\n";
       cout<<"PRESS 1: TO PRINT CHESSBOARD REPRESENTATION OF ANY TUPLE\n";
       cout<<"PRESS 2: TO EXIT\n";
       cout<<"ENTER YOUR CHOICE: ";
       cin>>choice;

        switch(choice)
        {
            case 1:printChessboard();break;
            case 2: return 0;
        }

    }


}
```

OUTPUT

```
ENTER VALUE OF N: 8

      POSSIBLE SOLUTIONS  {X1,X2,X3,X4,X5,X6,X7,X8}

   1| {1,5,8,6,3,7,2,4}         2| {1,6,8,3,7,4,2,5}
   3| {1,7,4,6,8,2,5,3}         4| {1,7,5,8,2,4,6,3}
   5| {2,4,6,8,3,1,7,5}         6| {2,5,7,1,3,8,6,4}
   7| {2,5,7,4,1,8,6,3}         8| {2,6,1,7,4,8,3,5}
   9| {2,6,8,3,1,4,7,5}        10| {2,7,3,6,8,5,1,4}
  11| {2,7,5,8,1,4,6,3}        12| {2,8,6,1,3,5,7,4}
  13| {3,1,7,5,8,2,4,6}        14| {3,5,2,8,1,7,4,6}
  15| {3,5,2,8,6,4,7,1}        16| {3,5,7,1,4,2,8,6}
  17| {3,5,8,4,1,7,2,6}        18| {3,6,2,5,8,1,7,4}
  19| {3,6,2,7,1,4,8,5}        20| {3,6,2,7,5,1,8,4}
  21| {3,6,4,1,8,5,7,2}        22| {3,6,4,2,8,5,7,1}
  23| {3,6,8,1,4,7,5,2}        24| {3,6,8,1,5,7,2,4}
  25| {3,6,8,2,4,1,7,5}        26| {3,7,2,8,5,1,4,6}
  27| {3,7,2,8,6,4,1,5}        28| {3,8,4,7,1,6,2,5}
  29| {4,1,5,8,2,7,3,6}        30| {4,1,5,8,6,3,7,2}
  31| {4,2,5,8,6,1,3,7}        32| {4,2,7,3,6,8,1,5}
  33| {4,2,7,3,6,8,5,1}        34| {4,2,7,5,1,8,6,3}
  35| {4,2,8,5,7,1,3,6}        36| {4,2,8,6,1,3,5,7}
  37| {4,6,1,5,2,8,3,7}        38| {4,6,8,2,7,1,3,5}
  39| {4,6,8,3,1,7,5,2}        40| {4,7,1,8,5,2,6,3}
  41| {4,7,3,8,2,5,1,6}        42| {4,7,5,2,6,1,3,8}
  43| {4,7,5,3,1,6,8,2}        44| {4,8,1,3,6,2,7,5}
  45| {4,8,1,5,7,2,6,3}        46| {4,8,5,3,1,7,2,6}
  47| {5,1,4,6,8,2,7,3}        48| {5,1,8,4,2,7,3,6}
  75| {6,3,7,4,1,8,2,5}        76| {6,4,1,5,8,2,7,3}
  77| {6,4,2,8,5,7,1,3}        78| {6,4,7,1,3,5,2,8}
  79| {6,4,7,1,8,2,5,3}        80| {6,8,2,4,1,7,5,3}
  81| {7,1,3,8,6,4,2,5}        82| {7,2,4,1,8,5,3,6}
  83| {7,2,6,3,1,4,8,5}        84| {7,3,1,6,8,5,2,4}
  85| {7,3,8,2,5,1,6,4}        86| {7,4,2,5,8,1,3,6}
  87| {7,4,2,8,6,1,3,5}        88| {7,5,3,1,6,8,2,4}
  89| {8,2,4,1,7,5,3,6}        90| {8,2,5,3,1,7,4,6}
  91| {8,3,1,6,2,5,7,4}        92| {8,4,1,3,6,2,7,5}
STEP COUNT: 82304



_____________________OPTIONS_____________________
PRESS 1: TO PRINT CHESSBOARD REPRESENTATION OF ANY TUPLE
PRESS 2: TO EXIT
ENTER YOUR CHOICE: 1
ENTER INDEX NUMBER OF THE TUPLE TO BE PRINTED: 92

^ ^ ^ ^ ^ ^ ^ Q
^ ^ ^ Q ^ ^ ^ ^
Q ^ ^ ^ ^ ^ ^ ^
^ ^ Q ^ ^ ^ ^ ^
^ ^ ^ ^ ^ Q ^ ^
^ Q ^ ^ ^ ^ ^ ^
^ ^ ^ ^ ^ ^ Q ^
^ ^ ^ ^ Q ^ ^ ^


_____________________OPTIONS_____________________
PRESS 1: TO PRINT CHESSBOARD REPRESENTATION OF ANY TUPLE
PRESS 2: TO EXIT
ENTER YOUR CHOICE:
```

1)

```
#include<iostream>
#include<iomanip>
#define CNT cntt++
using namespace std;
int n,m,cntt=0;

void whitespace(int x)
{
      for(int i=0;i<x;i++)
      cout<<" ";
}

void print_solTup(int elements[],int solTup[], int k)
{

      int i;
      cout<<"{";
      for(i=1;i<=k;i++)
      {
            cout<<solTup[i];CNT;
        if(i!=k)
        cout<<",";
      }
      cout<<"} ";

      //i represents no. of spaces elements occupied on the screen
      whitespace(30-((i*2)+1));  //x2  No 0f elements,commas and brackets
combined
    cout<<"[";
      for(i=1;i<=k;i++)
      {
            if(solTup[i]==1)
        cout<<elements[i];
         if(i!=k&&solTup[i]==1)
        cout<<",";
      }
      cout<<"] ";

      cout<<endl;
}

void sum_of_subsets(int elements[],int solTup[],int s, int k, int r)
{
      if(k>n)
      return;

      solTup[k]=1;CNT;

      if(s+elements[k]==m)
      {CNT;print_solTup(elements,solTup,k);CNT;}
      else if((s+elements[k]+elements[k+1])<=m)
      {CNT;sum_of_subsets(elements,solTup,s+elements[k],k+1,r-
elements[k]);CNT;}

      if((s+r-elements[k]>=m) && (s+elements[k+1]<=m))
      {
            CNT;
            solTup[k]=0;CNT;
            sum_of_subsets(elements,solTup,s,k+1,r-elements[k]);CNT;
      }
}
```

```
int main()
{
	int r=0,i;
	cout<<"ENTER THE SIZE OF THE SET"<<endl;
	cin>>n;

	int elements[n+1];
	int solTup[n+1];

	cout<<"ENTER ELEMENTS INTO YOUR ARRAY:\n";
	for(i=1;i<=n;i++)
	{
		cin>>elements[i];
		solTup[i]=0;
		r+=elements[i];
	}

	cout<<"ENTER TARGET SUM: ";
	cin>>m;cout<<endl;


    cout<<std::left<<setw(30)<<"TUPLE"<<std::left<<setw(30)<<"ELEMENTS:
"<<endl;
	sum_of_subsets(elements,solTup,0,1,r);
	cout<<"\nSTEP COUNT: "<<cntt<<endl;
	return 0;
}
```

OUTPUT

```
ENTER THE SIZE OF THE SET
4
ENTER ELEMENTS INTO YOUR ARRAY:
1 3 4 5
ENTER TARGET SUM: 8

TUPLE                          ELEMENTS:
{1,1,1}                       [1,3,4]
{0,1,0,1}                     [3,5]

STEP COUNT: 37
PS C:\Users\Lloyd\Desktop\MADF\EXPT 13>
```

1)

```
#include<iostream>
#include<windows.h>
using namespace std;
#define MAX 10

#define NEXTLINE cout<<"\n"
#define CNT count++

int GRAPH[MAX][MAX],Colors[MAX];
int V,E,M;
int count=0;

enum ColourScheme{BLUE=1,GREEN,CYAN,RED,PURPLE,YELLOW,};

void Line_Generator(int n)
{
      int i;
      for(i=0;i<n;i++)
      cout<<"-";
}

void G_create()
{
    int i,j;
    cout<<"ENTER NUMBER OF VERTICES:\n";
    cin>>V;

       cout<<"ENTER NUMBER OF COLOURS:\n";
    cin>>M;

    cout<<"ENTER ADJACENCY MATRIX:\n";

    for(i=0;i<V;i++)
    for(j=0;j<V;j++)
    {
        cin>>GRAPH[i][j];
    }

    for(i=0;i<V;i++)
    Colors[i]=0;
}

void SetColor(int value){
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),  value);
}

void  print_Coloured_Graph()
{
    int i;
    for(i=0;i<V;i++)
    {
        if(Colors[i]==1)
        SetColor(RED);

         if(Colors[i]==2)
           SetColor(BLUE);

         if(Colors[i]==3)
           SetColor(GREEN);
```

```
          if(Colors[i]==4)
            SetColor(YELLOW);

          if(Colors[i]==5)
            SetColor(CYAN);

          if(Colors[i]==6)
            SetColor(PURPLE);

         cout<<Colors[i]<<"     ";
    }
    NEXTLINE;
}

void NextValue(int k)
{
    while(1)
    {
         Colors[k]=(Colors[k]+1)%(M+1);CNT;
         if(Colors[k]==0)
         {CNT;return;}
         int j;
         for(j=0;j<V;j++)
         {
            if(GRAPH[k][j]!=0 && Colors[j]==Colors[k])
                 {CNT;break;}
         }
         if(j==V)
        {CNT;return;}
    }

}

void m_coloring(int k)
{
    while(1)
    {
        CNT;NextValue(k);CNT;
        if(Colors[k]==0)
        {CNT;return;}

        if(k==V-1)
        {CNT;print_Coloured_Graph();}
        else
        { m_coloring(k+1);CNT;}
    }
}

int main()
{
    G_create();
    cout<<"\n\n****** |COLOURED GRAPH| ******\n";
    cout<<"------------------------------\n";
    cout<<"Red|Blue|Green|Yellow|Cyan|Purple\n";
    cout<<" 1 |  2 |  3  |   4  |  5 |  6   \n\n";

    for(int i=0;i<V;i++)
    {cout<<"["<<i+1<<"]"<<"   ";}
    cout<<endl;
    Line_Generator((V*6)+3);
    cout<<endl;

    m_coloring(0);
```

```
    cout<<"STEP COUNT: "<<count<<endl;
}
```

OUTPUT

```
ENTER NUMBER OF VERTICES:
5
ENTER NUMBER OF COLOURS:
3
ENTER ADJACENCY MATRIX:
0 1 1 1 0
1 0 0 0 1
1 0 0 1 1
1 0 1 0 1
0 1 1 1 0


****** |COLOURED GRAPH| ******
------------------------------
Red|Blue|Green|Yellow|Cyan|Purple
 1 |  2 |  3  |   4  |  5 |  6

[1]   [2]   [3]   [4]   [5]
--------------------------------
1     2     2     3     1
1     2     3     2     1
1     3     2     3     1
1     3     3     2     1
2     1     1     3     2
2     1     3     1     2
2     3     1     3     2
2     3     3     1     2
3     1     1     2     3
3     1     2     1     3
3     2     1     2     3
3     2     2     1     3
STEP COUNT: 509
PS C:\Users\Lloyd\Desktop\MADF\EXPT 14>
```

1)

```
#include<iostream>
#include<windows.h>
#define RED 4
#define WHITE 7
#define CNT cntt++
using namespace std;
int m,n,cntt=0;

void SetColor(int value){
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),  value);
}

void HighlightPattern(int i,string text,string pattern)
{
     cout<<"\n\nPATTERN MATCHED!"<<endl;
    //substr() used to slice the string for colouring
    cout<<text.substr(0,i);
    SetColor(RED);
    cout<<text.substr(i,m);
    SetColor(WHITE);
    cout<<text.substr(i+m,n);

    cout<<endl<<"FROM: "<<i+1<<" TO "<<i+m;
}

int BruteForce_patternMatching(string text, string pattern)
{
    int j;
    for(int i=0;i<=n-m;i++)
    {
        CNT;
        j=0;   CNT;
        while(j<m && text[i+j]==pattern[j])
        {
              CNT; j++;  CNT;
        }
        if(j==m)
        {   CNT;return i;}
    }

    cout<<"NO SUBSTRING PRERSENT!";   CNT;
      cout<<"\nSTEP COUNT: "<<cntt;
    exit(0);

}

int main()
{
    string text,pattern;int i;

    cout<<"ENTER A STRING: ";
    getline(cin,text);
    n=text.length();

    cout<<"ENTER PATTERN STRING: ";
    getline(cin,pattern);
    m=pattern.length();


HighlightPattern(BruteForce_patternMatching(text,pattern),text,pattern);
```

```
    cout<<"\nSTEP COUNT: "<<cntt;
}
```

OUTPUT

```
ENTER A STRING: abacaabaccabacabaabb
ENTER PATTERN STRING: abacab


PATTERN MATCHED!
abacaabaccabacabaabb
FROM: 11 TO 16
STEP COUNT: 59
PS C:\Users\Lloyd\Desktop\MADF\EXPT 15>
```

```
ENTER A STRING: LLOYD ALRICH COSTA
ENTER PATTERN STRING: YD ALR


PATTERN MATCHED!
LLOYD ALRICH COSTA
FROM: 4 TO 9
STEP COUNT: 21
PS C:\Users\Lloyd\Desktop\MADF\EXPT 15>
```

2)

```
#include<iostream>
#include<windows.h>
#include<algorithm>
#include<string>
#define CNT cntt++
#define RED 4
#define WHITE 7
using namespace std;
int m,n,cntt=0;

void SetColor(int value){
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),  value);
}

void HighlightPattern(int i,string text,string pattern)
{
     cout<<"\n\nPATTERN MATCHED!"<<endl;
    //substr() used to slice the string for colouring
    cout<<text.substr(0,i);
    SetColor(RED);
    cout<<text.substr(i,m);
    SetColor(WHITE);
    cout<<text.substr(i+m,n);

    cout<<endl<<"FROM: "<<i+1<<" TO "<<i+m;
}
int last(string pattern,char c)
{
    int L=pattern.find_last_of(c);
    if(L!=string::npos)
    return L;
    else
    return -1;
}

int BMmatch(string text, string pattern)
{
    int i=m-1;CNT;
    int j=m-1;CNT;

    do{
        CNT;
        if(pattern[j]==text[i])
        {
            CNT;
            if(j==0)
            {CNT;return i;}
            else
            {
                i--;CNT;
                j--;CNT;
            }
        }
        else
        {
            i=i+m-min(j,1+last(pattern,text[i]));CNT;
            j=m-1;CNT;
        }
    }while(i<=n-1);
```

```
    cout<<"NO SUBSTRING PRERSENT!";
     cout<<"\nSTEP COUNT: "<<cntt;
    exit(0);
}

int main()
{
    string text,pattern;int i;

    cout<<"ENTER A STRING: ";
    getline(cin,text);
    n=text.length();

    cout<<"ENTER PATTERN STRING: ";
    getline(cin,pattern);
    m=pattern.length();

    HighlightPattern(BMmatch(text,pattern),text,pattern);
     cout<<"\nSTEP COUNT: "<<cntt;
}
```

OUTPUT

```
ENTER A STRING: abacaabadcabacabaabb
ENTER PATTERN STRING: abacab


PATTERN MATCHED!
abacaabadcabacabaabb
FROM: 11 TO 16
STEP COUNT: 48
PS C:\Users\Lloyd\Desktop\MADF\EXPT 15>
```

```
ENTER A STRING: LLOYD COSTA
ENTER PATTERN STRING: OYD COS


PATTERN MATCHED!
LLOYD COSTA
FROM: 3 TO 9
STEP COUNT: 32
PS C:\Users\Lloyd\Desktop\MADF\EXPT 15>
```

1)

```
#include<iostream>
#include<windows.h>
#include<algorithm>
#include<string>
#define CNT cntt++
#define RED 4
#define WHITE 7
using namespace std;
int m,n,f[200],cntt=0;


void SetColor(int value){
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),  value);
}

void HighlightPattern(int i,string text,string pattern)
{
     cout<<"\n\nPATTERN MATCHED!"<<endl;
    //substr() used to slice the string for colouring
    cout<<text.substr(0,i);
    SetColor(RED);
    cout<<text.substr(i,m);
    SetColor(WHITE);
    cout<<text.substr(i+m,n);

    cout<<endl<<"FROM: "<<i+1<<" TO "<<i+m;
}

void KMPFailureFunction(string pattern)
{
    int i=1,j=0;
    f[0]=0;CNT;
    while(i<m)
    {
        CNT;
        if(pattern[j]==pattern[i])
        {
            CNT;
            f[i]=j+1;CNT;
            i++;CNT;
            j++;CNT;
        }
        else if(j>0)
        {
            j=f[j-1];CNT;
        }
        else
        {
            f[i]=0;CNT;
            i++;CNT;
        }
    }

}

int KMPmatch(string text, string pattern)
{
    KMPFailureFunction(pattern);
    int i=0,j=0;CNT;
    while(i<n)
```

```
    {
        CNT;
        if(pattern[j]==text[i])
        {
            if(j == m-1)
            {
               CNT; return i-m+1;
            }
            i++;j++;CNT;CNT;
        }
        else if(j>0)
        {
            j=f[j-1];CNT;
        }
        else
        {
            i++;CNT;
        }
    }
    cout<<"NO SUBSTRING PRERSENT!"; CNT;
      cout<<"\nSTEP COUNT: "<<cntt;
    exit(0);

}

int main()
{
    string text,pattern;int i;

    cout<<"ENTER A STRING: ";
    getline(cin,text);
    n=text.length();

    cout<<"ENTER PATTERN STRING: ";
    getline(cin,pattern);
    m=pattern.length();

    HighlightPattern(KMPmatch(text,pattern),text,pattern);
      cout<<"\nSTEP COUNT: "<<cntt;
}
```

OUTPUT

```
ENTER A STRING: abacaabaccabacabaabb
ENTER PATTERN STRING: abacab


PATTERN MATCHED!
abacaabaccabacabaabb
FROM: 11 TO 16
STEP COUNT: 77
PS C:\Users\Lloyd\Desktop\MADF\EXPT 16>
```

```
ENTER A STRING: LLOYD COSTA
ENTER PATTERN STRING: YD COST


PATTERN MATCHED!
LLOYD COSTA
FROM: 4 TO 10
STEP COUNT: 46
PS C:\Users\Lloyd\Desktop\MADF\EXPT 16>
```