int find(int x) { return pre[x] == x ? x : pre[x] = find(pre[x]);}

//并查集的find

// 匈牙利算法 二分图最大匹配

int map[105][105];

int visit[105],flag[105];

int n,m;

bool dfs(int a)

{

for(int i=1; i<=n; i++)

{

if(map[a][i]&&!visit[i])

{

visit[i]=1;

if(flag[i]==0||dfs(flag[i]))

{

flag[i]=a;

return true;

}

}

}

return false;

}

int main()

{

while(cin>>n1 >>n2 >>m)

{

memset(map,0,sizeof(map));

for(int i=1; i<=m; i++)

{

int x,y;

cin>>x>>y;

map[x][y]=1;

}

memset(flag,0,sizeof(flag));

int result=0;

for(int i=1; i<=n1; i++)

{

memset(visit,0,sizeof(visit));

if(dfs(i))result++;

}

cout<<result<<endl;

}

return 0;

}

bool find(int x){

int i,j;

for (j=1;j<=m;j++){ //扫描每个妹子

if (line[x][j]==true && used[j]==false)

//如果有暧昧并且还没有标记过(这里标记的意思是这次查找曾试图改变过该妹子的归属问题，但是没有成功，所以就不用瞎费工夫了）

{

used[j]=1;

if (girl[j]==0 || find(girl[j])) {

//名花无主或者能腾出个位置来，这里使用递归

girl[j]=x;

return true;

}

}

}

return false;

}

//在主程序我们这样做：每一步相当于我们上面描述的一二三四中的一步

for (i=1;i<=n;i++)

{

memset(used,0,sizeof(used)); //这个在每一步中清空

if find(i) all+=1;

}