第一題

(a) 因為分子是15的階層，用combiBad(n, m)會把fact(n)算出才除分母，分子會overflow。

(b)

```
#include <iostream>
using namespace std;

int multi(int s, int t)
{
    int prod = 1;
    for(int i = s; i <= t; i++)
    prod *= i;
    return prod;
}
int combiRep(int n, int m)
{
          if(2 * m > n)
                return multi( m+1,n ) / multi(1 , m-n);
          else
                return multi( n-m+1,n ) / multi( 1,m );
}

int main() {
          int n,m;
          cin>>n>>m;
          cout<<combiRep(n, m);
          // your code goes here
          return 0;
}
```

第二題

(a)

```
int combiRec(int n, int m)
{
        if(n<m) return -1 ;
        if(m==1) return n;
        if(n==m) return 1;
        return combiRec(n-1,m) + combiRec(n-1,m-1);
}
```

(b) 佩蓉的方法還是有用到多個數字乘了再除的步驟，但是昱賢的方法會在解子問題的時候只有用到加法，所以不會有乘過多數字造成overflow的問題。

(c) 因為遞迴的方法會造成子問題重複計算很多次，如果是敬傑或是佩蓉的方法是linear time，每個數字乘過一次後就不會再用到那個數字；但是遞迴時如果C(n,m)呼叫C(n-1,m)、C(n-1,m-1)

，而C(n,m+1)也會呼叫C(n-1,m+1)、C(n-1,m)，C(n-1,m)被呼叫兩次但是第一次的結果無法傳遞給第二次，所以必須解同樣的子問題，造成時間複雜度提高。