กรณี recursive

```
#include <stdio.h>
#include <math.h>

void A(int x)
{
   if( x == 0 )                  → เงื่อนไขหยุด
   {
      return;
   }
   else
   {
      printf("%d\n",x);
      x = x - 1;                 → เปลี่ยนแปลง
      A(x);                      → เปลี่ยนแปลง
   }
}

int main()
{
   A(5);                         → เริ่มต้น 5
   return 0;
}
```

ตัวอย่างสมการ

$a_n = 2a_{n-1} + 5$ จากสมการนี้ เขียน code ออกมาเป็น

$a_0 = 1$

```
int _stop = 0;
void B(int x,int y)   //x = round, y = result
{
    if( _stop == x )
    {
        printf("a=%d : %d\n",x,y);
        return;
    }
    else
    {
        printf("a=%d : %d\n",x,y);
        y = (2*y) + 5;
        x = x + 1;
        B(x,y);
    }
}
int main()
{
    _stop = 5;
    B(0,1);
    return 0;
}
```

5) กรณีแสดงผลข้อมูล

```
#include <stdio.h>
#include <math.h>
int _stop = 0;
void B(int x,int y)   //x = round, y = result
{
    if( _stop == x )
    {
        return;
    }
    else
    {
        y = (2*y) + 5;
        x = x + 1;
        B(x,y);
        printf("a=%d : %d\n",x,y);
    }
}
int main()
{
    _stop = 5;
    B(0,1);
    return 0;
}
```

ไม่ใช้ recursive

```
int A[3][3],B[3][3],C[3][3],D[3][3];
void myprint(int A[3][3])
{
	int i=0,j=0;
	for(i=0;i<3;i++)
	{
		for(j=0;j<3;j++)
		{
			printf("%d ",A[i][j]);
		}
		printf("\n");
	}
	printf("-----------------------------\n");
}
void copy(int A[3][3],int B[3][3]) //Copy A To B
{
	int i=0,j=0;
	for(i=0;i<3;i++)
	{
		for(j=0;j<3;j++)
		{
			B[i][j] = A[i][j];
		}
	}
}
int main()
{
	int i=0,j=0,x=0,y=0,ii=0,jj=0;
	A[0][0] = 2;	A[0][1] = 1;	A[0][2] = 2;
	A[1][0] = 1;
	A[2][0] = 2;
	myprint(A);
//-------------------------------------------------
	copy(A,B);
	for(x=0;x<3;x++)
	{
		for(y=0;y<3;y++)
		{
			if(B[x][y] == 0)
			{
				B[x][y] = 1;
				printf("....L1\n"); myprint(B);
				//-------------------------------------------------
				copy(B,C);
				for(i=0;i<3;i++)
				{
					for(j=0;j<3;j++)
					{
						if(C[i][j] == 0)
						{
							C[i][j] = 2;
							printf("....L2\n"); myprint(C);
							//-------------------------------------------------
							copy(C,D);
							for(ii=0;ii<3;ii++)
							{
								for(jj=0;jj<3;jj++)
								{
									if(D[ii][jj] == 0)
									{
										D[ii][jj] = 1;
										printf("....L3\n"); myprint(D);
										//-------------------------------------------------
										//.................
										//-------------------------------------------------
										copy(C,D);
									}
								}
							}
							//-------------------------------------------------
							copy(B,C);
						}
					}
				}
				//-------------------------------------------------
				copy(A,B);
			}
		}
	}
//-------------------------------------------------
	return 0;
}
```

ใช้ recursive

```
#include <stdio.h>
#include <math.h>
void myprint(int A[3][3])
{
    int i=0,j=0;
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            printf("%d ",A[i][j]);
        }
        printf("\n");
    }
    printf("-------------------------\n");
}
void copy(int A[3][3],int B[3][3]) //Copy A To B
{
    int i=0,j=0;
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            B[i][j] = A[i][j];
        }
    }
}
void play(int C[3][3],int turn)
{
    int ii=0,jj=0,D[3][3];
    int have = 0;
    for(ii=0;ii<3;ii++)
    {
        for(jj=0;jj<3;jj++)
        {
            if(C[ii][jj] == 0)
            {
                have = 1; break;
            }
        }
        if(have){ break; }
    }
    if(have==0){ printf("\n------- Finish game -------\n"); return; }
    //====================
    copy(C,D);
    for(ii=0;ii<3;ii++)
    {
        for(jj=0;jj<3;jj++)
        {
            if(D[ii][jj] == 0)
            {
                D[ii][jj] = turn;
                myprint(D);
                //====================
                if(turn == 2) { turn = 1; }
                else          { turn = 2; }
                //====================
                play(D,turn);
                copy(C,D);
            }
        }
    }
    //====================
}

int A[3][3],B[3][3],C[3][3],D[3][3];
int main()
{
    A[0][0] = 2;   A[0][1] = 1;   A[0][2] = 2;
    A[1][0] = 1;
    A[2][0] = 2;
    myprint(A);
    play(A,1);
    return 0;
}
```

1) กรณีค้นหาคำตอบแบบแตกเป็น Tree

```
int Fibonacci(int num)
{
   if(num == 0)          {   return 0;   }
   else if(num == 1)     {   return 1;   }
   else                  {   return( Fibonacci(num-1) + Fibonacci(num-2) );   }
}
cout<<Fibonacci(10)<<endl;
```

2) กรณีค้นหาคำตอบแบบแตกเป็น 2 ทางในรูป binary

```
int ans[5] = {0,0,0,0,0};
void brute(int j)
{
   if (j == size)   { for(int i=0 ; i < size ; i++) {   cout<<ans[i]<<" ";  }   cout<<endl; }
   else
   {
      int num = j+1;
      ans[j] = 0;
      brute(num);
      ans[j] = 1;
      brute(num);
   }
}
brute(0);
```

3) กรณีสลับลำดับทั้งหมดพวกนี้เป็น PERMUTATION พวกนี้ n!

```
int size = 5;
int Array[100];
void permute(int j)
{
   if (j == size)
   {
         for(int i=0 ; i < size ; i++)   {   cout<<Array[i]<<" ";   }   cout<<endl;
   }
   else
   {
      for (int i = j; i < size; i++)
      {
          int T = Array[j];
          Array[j] = Array[i];
             Array[i] = T;
          permute(j+1);
             T = Array[j];
          Array[j] = Array[i];
             Array[i] = T;
      }
   }
}
for(int i=0;i<100;i++){ Array[i] = i+1;}
permute(0);
```

4) กรณีแตกหารข้อมูลออก

```
Function merge (l, i, j, r)
{
   ขั้นตอนการทำงานของฟังก์ชัน merge
}
Function split (i, j)
{
   ขั้นตอนการทำงานของฟังก์ชัน split
   m = (i+j)/2
   split (i, m)
   split (m+1, j)
   merge ( i, m, m+1, j )
}
split (0, s)
```

กรณีแตกหารข้อมูลออก

```
#include<bits/stdc++.h>
using namespace std;
int max_ = 0;
int nums[] = { 2, 7, 9, 3, 1, 6, 7, 8, 4 };
void findMax(int low, int high)
{
    if (low == high)
    {
        if ( max_ < nums[low] ) {  max_ = nums[low];  }
        return;
    }
    if (high - low == 1)
    {
            if (max_ < nums[high])   {  max_ = nums[high]; }
            if (max_ < nums[low])    {  max_ = nums[low];  }
        return;
    }
    int mid = (low + high) / 2;
    findMax( low, mid );
    findMax( mid + 1, high);
}
int main()
{
    findMax(0, 8);
    cout << max_ << endl;
    return 0;
}
```