实验一：

1. 题目：给定整数n所构造的素数环是指，用1~n共n个不重复整数构造一个环形结构，使相邻两个数之和为素数。例如6的素数环为1，4，3，2，5，6。编写非递归函数void primeCycle(int n)，由用户输入整数n，判断是否能够产生素数环，并输出结果。
2. 思路及代码：

#include <stdio.h>

#include <malloc.h>

#include <string.h>

#include <math.h>

int isPrime(int n) //判断是否为素数

{

int i,k;

k = (int)sqrt(n);

for(i=2;i<=k;i++)

{

if(n%i==0)

return 0;

}

return 1;

}

int canPlace(int \*p,int i,int j,int n) //传入数组首指针，要放置的位置，要放置的数值，数组大小，函数用来检查能不能放入该数

{

int k,flag;

for(k=0;k<i;k++) //检查前面是否已经使用过该数

{

if(p[k]==j)

return 0;

}

flag = isPrime(j+p[i-1]); //若放置则与前数和是否为素数

if(flag==1 && i==n-1) //若是最后一个位置则需要和第一个数构成素数环

{

flag = isPrime(j+p[0]);

}

return flag;

}

void showResult(int \*p,int n) //传入数组首地址和数组大小

{

int i;

static count = 0;

printf("\*\*\*\*第%d个解\*\*\*\*\n",++count);

for(i=0;i<n;i++)

{

printf("%d ",p[i]);

}

printf("\n");

}

void primeCycle(int n)

{

int i=1,j,k;

int \*p = (int \*)malloc(sizeof(int)\*n); //动态开辟数组空间

memset(p,0,n\*sizeof(int)); //数组置零

p[0] = 1; //先给第0号位赋上值 ，因为是环状结构，为避免出现相同结构的解，将第一个元素固定。

while(i>=1)

{

for(j=p[i]+1;j<=n;j++) //找出能在该位置放置的满足条件的第一个数。如果是第一次寻找，则j从1开始，若是回溯，则从上次 确定的值开始

{

if(canPlace(p,i,j,n)==1)

{

p[i] = j;

break;

}

}

if(j>n) //如果上一个循环正常结束，即该位置没有可放置的数

{

p[i--] = 0; //p[i]=0;i--;回溯至上一个位置,这个位置置0。

}

else if(i==n-1) //如果最后一个位置的数也放置完毕，则输出结果，并将前一个数置零以便寻找下一种解情况

{

showResult(p,n);

p[i--] = 0;

}

else //如果还没有放置完数，则将i++继续进行大循环。

i++;

}

free(p); //释放空间

printf("finished\n");

}

void main()

{

int n;

printf("Input an integer\n");

scanf("%d",&n);

primeCycle(n);

}

（3）