1.

//201233311 백지한

#include < stdio.h>

#include < stdlib.h>

struct POINT

{

float x;

float y;

int flag;

};

float angle ( struct POINT, struct POINT );

void calcualte( struct POINT \*);

struct POINT convex[100];

void main()

{

int num;

struct POINT \*p;

int i,j;

int k;

int cnt = 1;

float\*ca;

int min = INT\_MAX;

int index = 0;

printf("Input the num : ");

scanf("%d", &num);

p = (struct POINT \*)malloc ( sizeof(struct POINT) \* num);

ca = (float \*) malloc ( sizeof( float ) \* num);

//get the value

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

{

scanf("%f", & p[i].x);

scanf("%f", & p[i].y);

p[i].flag = 0;

}

convex[0].x = p[0].x;

convex[0].y = p[0].y;

p[0].flag = 1;

//this will find the next anchor point and keep wrapping the gift box.

for(i = 0; i < num-1;i++)

{

for( j = i; j < num-1; j++)

{

if( i != j && p[j].flag == 0)

ca[j] = angle( p[cnt-1],p[j]);

else

ca[j] = 9999999;

//printf("%.2f\n", ca[j]);

}

for( k = 0; k < num; k++)

{

printf("%d ", ca[k]);

}

for( k = 0; k < num; k++)

{

if( ca[k] < min )

{

min = ca[k];

index = k;

}

}

//printf("ind %d\n", index);

//printf("min : %f\n", min);

//this will put the value in the convex

convex[cnt].x = p[index].x;

convex[cnt].y = p[index].y;

p[index].flag = 1;

cnt++;

//this will stop the while loop

if( (convex[0].x == convex[cnt-1].x )&& (convex[0].y == convex[cnt-1].y))

break;

//printf("%d\n", cnt);

min = 99999999;

}

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

{

printf("%f %f\n", convex[i].x, convex[i].y);

}

//calculate(convex);

//calculate ( p);

}

//this will compute the raltive angle

void calculate()

{

// in this function it will calculate the polygons total space to compare how much space is wasted

// using x1, y1 , x2, y2 ..........

}

float angle(struct POINT a, struct POINT b)

{

float dx;

float dy;

float angle = 0;

dx = b.x - a.x;

dy = b.y - a.y;

if( dx >= 0 && dy == 0)

{

angle = 0;

}

else

{

angle = abs( dx ) / (abs(dx+dy));

if( dx < 0 && dy >= 0)

angle = angle - 2;

else if( dx <= 0 && dy < 0)

angle = angle +2;

else if( dx > 0 && dy < 0 )

angle = angle - 4;

}

angle = angle \* 90;

return angle;

}

2.

#include < stdio.h>

#include < stdlib.h>

void rsa(int , int ,int);

int mod( int , int, int);

int odd( int);

int euclid( int, int );

int cypher(int, int, int);

void decypher( int, int, int );

void main()

{

int p;

int q;

int msg;

printf("Input p: ");

scanf("%d", &p);

printf("Input q: ");

scanf("%d", &q);

printf("Input msg : ");

scanf("%d", &msg);

rsa( p,q,msg);

}

void rsa( int p, int q, int msg)

{

int n;

int pin;

int w;

int e = 3;

int d;

int re;

int cnt =1;

n = p\*q;

pin = (p-1)\*(q-1);

//printf("%d\n", euclid(9,24));

//public key e

//this will find the key value e

while(1)

{

if( (odd(e) == 0) &&( euclid(e, pin)== 1))

break;

else

e++;

}

//e = 5;

//to find the secret key

while(1)

{

if( ((pin \* cnt) + 1) % e == 0)

{

d = (pin\*cnt +1) / e;

break;

}

cnt++;

}

printf("public %d %d \n secret %d %d\n", e, n, d, n);

//scipher

re = cypher( e, n, msg);

//decipher the value

decypher(re, d, n);

}

// to determine if two nums are ralative prime.

int euclid( int a, int b)

{

if( a % b == 0)

return b;

else return euclid(b, a%b);

}

//cipher the message

int cypher( int e, int n, int msg)

{

int c;

c= mod( msg, e, n);

printf("cipher : %d\n", c);

return c;

}

//dcipher the message

void decypher( int c, int d, int n)

{

int m ;

m = mod( c,d,n);

printf("original : %d", m);

}

//to determine if odd

int odd( int a )

{

if ( a % 2== 0)

return 1;

else

return 0;

}

//this is a function of exponentional modular

int mod( int a, int b, int n)

{

int c;

int d;

int bit = INT\_MAX;

int i;

bit = bit - ( bit >> 1);

c = 0;

d = 1;

for( i = 0; i < 31; i++)

{

c = c \* 2;

d = (d \* d ) % n;

if( (b & bit )!= 0)

{

c++;

d = (d\*a) % n;

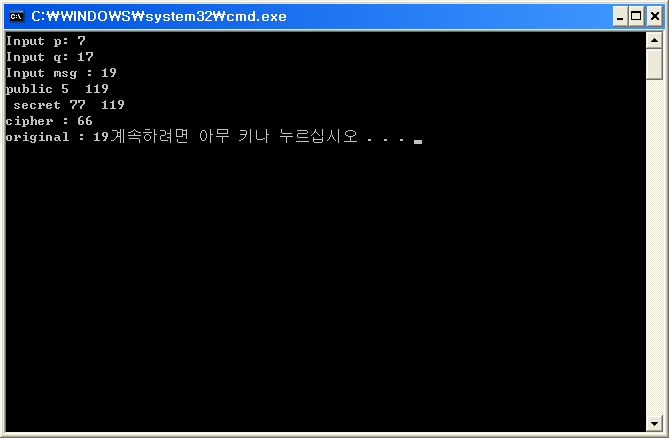
}

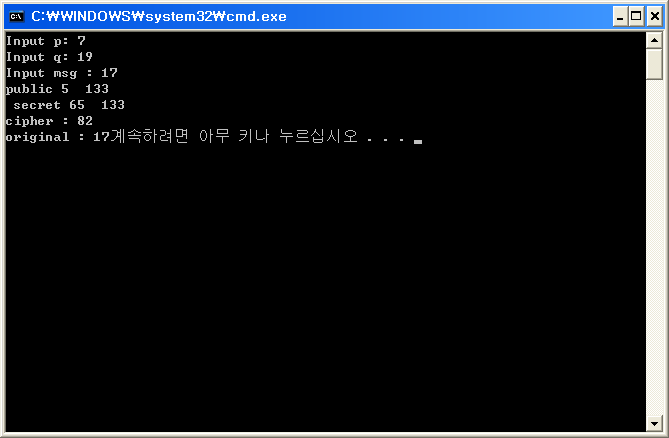
bit = bit >> 1;

}

return d;

}





3.

Back tracking algorithm is a algorithm that finds the possible solution. It finds the solution by determining if the value is promising or not promising,

If the value if promising than it keeps going and if it is not promising than goes back, and for the non promising nodes we cut the nodes and it is called pruning,

So if the nodes are pruned than it means it is a dead so don’t have to look the way again.

It reduces a time for getting a solution when using a backtracking algorithms.

4.

Main

{

If( status == undergraduate )

Cheat = no;

Else

Number of course take( );

}

Number of course( )

{

If( num < 10)

averagegpa( );

Else

Cheat = no;  
}

Averagegpa( )

{

If( gpa > 3)

Cheat = no;

Else

cheat = yes;

}

5.

Np – completeness means it cant’ be finished in a polynomial time. It means It needs exponential time .

Decision problem means it is answered by yes or no, ( deciding yes or no ) and optimization problem is finding a optimal solution among many other solutions.