|  |  |
| --- | --- |
| in.txt | Out.txt |
| int gcd(int u, int v) // calculate the gcd of u and v  {  if (v == 0) return u;  else return gcd(v,u-u/v\*v); /\* v,u-u/v\*v is equals to u mod v\*/  }  /\* 123123  123123  123123  123123 \*\*\*\*  \*\*/  void main(void)  {  int x; int y; double temp;  int a[100];  temp = 1.1;  for(i = 1; i < 1; ) {  for(j = 2; ; j=j+1)  a[1+1] = b;  }  x = input();  y = input();  if(x < y)  {  temp = x;  x = y;  y = temp;  }  output(gcd(x,y));  while(y > x) {  a[b=x+y-1] = b = c;  }  return 0;  } | Fun: gcd  Type: int  Param:  Type: int  Id: u  Param:  Type: int  Id: v  Compound: {  Selection: if  Op: ==  Id: v  Const: 0  Return:  Id: u  else:  Return:  Fun: gcd  Id: v  Op: -  Id: u  Op: \*  Op: /  Id: u  Id: v  Id: v  }  Fun: main  Type: void  Param:  Type: void  Compound: {  Declaration:  Type: int  Id: x  Declaration:  Type: int  Id: y  Declaration:  Type: double  Id: temp  Declaration:  Type: int  Array:  Id: a  Const: 100  Expression:  Op: =  Id: temp  Const: 1.1  For: for  Op: =  Id: i  Const: 1  Op: <  Id: i  Const: 1  Empty:  Compound: {  For: for  Op: =  Id: j  Const: 2  Empty:  Op: =  Id: j  Op: +  Id: j  Const: 1  Expression:  Op: =  Array:  Id: a  Op: +  Const: 1  Const: 1  Id: b  }  Expression:  Op: =  Id: x  Fun: input  Expression:  Op: =  Id: y  Fun: input  Selection: if  Op: <  Id: x  Id: y  Compound: {  Expression:  Op: =  Id: temp  Id: x  Expression:  Op: =  Id: x  Id: y  Expression:  Op: =  Id: y  Id: temp  }  Expression:  Fun: output  Fun: gcd  Id: x  Id: y  Iteration: while  Op: >  Id: y  Id: x  Compound: {  Expression:  Op: =  Array:  Id: a  Op: =  Id: b  Op: +  Id: x  Op: -  Id: y  Const: 1  Op: =  Id: b  Id: c  }  Return:  Const: 0  } |