```
.686                            ; create 32 bit code
.model flat, C                  ; 32 bit memory model
 option casemap:none            ; case sensitive

.data                           ; start of data section
public g                        ; export of variable g
g DWORD 4                       ; declare global variable g initialised to 4

.code                           ; start of code section

;
; t1.asm
;
; Copyright (C) 2018 dooleyb1@tcd.ie
;

; function to calculate min(a, b, c)
;
; returns result in eax

public    min                   ; make sure function name is exported

min:      push  ebp             ; push frame pointer
          mov   ebp, esp        ; update ebp

          mov   eax, [ebp+8]    ; v = a
          mov   ecx, [ebp+12]   ; ecx = b
          cmp   ecx, eax        ; if (b < v)
          jge   min_1           ;
          mov   eax, ecx        ; v = b

min_1:    mov   ecx, [ebp+16]   ; ecx = c
          cmp   ecx, eax        ; if (c < v)
          jge   min_2           ;
          mov   eax, ecx        ; v = c

min_2:    mov   esp, ebp        ; restore esp
          pop   ebp             ; restore ebp
          ret   0               ; return

; function to calculate p(i, j, k, l)
;
; returns min(min(g, i, j), k, l) in eax

public    p                     ; make sure function name is exported

p:        push  ebp             ; push frame pointer
          mov   ebp, esp        ; update ebp

          push  [ebp+12]        ; push j onto stack
          push  [ebp+8]         ; push i onto stack
          push  g               ; push g onto stack

          call  min             ; eax = min(g, i, j)

          push  [ebp+20]        ; push l onto stack
          push  [ebp+16]        ; push k onto stack
          push  eax             ; push min(g, i, j) onto stack

          call  min             ; eax = min(min(g,i,j), k, l)

          mov   esp, ebp        ; restore esp
          pop   ebp             ; restore ebp
          ret   0               ; return
```

```
65
66  ; function to calculate gcd(a, b)
67  ;
68  ; returns gcd(a, b) in eax
69
70  public      gcd                 ; make sure function name is exported
71
72  gcd:      push  ebp             ; push frame pointer
73            mov   ebp, esp        ; update ebp
74
75            mov   eax, [ebp+12]   ; eax = b
76            cmp   eax, 0          ; if(b==0)
77            je    gcd_retA        ;  return a
78
79            mov   eax, [ebp+8]    ; eax = a (dividend)
80            cdq                   ; sign extend eax into edx
81            mov   ecx, [ebp+12]   ; ecx = b (divisor)
82            idiv  ecx             ; edx = a % b
83
84            push  edx             ; push edx (a % b) onto stack
85            push  [ebp+12]        ; push b onto stack
86
87            call  gcd             ; eax = gcd(b, (a % b))
88            jmp   gcd_done        ;
89
90  gcd_retA: mov eax, [ebp+8]      ; eax = a
91
92  gcd_done: mov     esp, ebp      ; restore esp
93              pop     ebp         ; restore ebp
94              ret     0           ; return
95
96  end
97
```