Q1:

Push ebp push ebp onto stack

Mov ebp, esp move esp’s data to ebp

And esp, 0FFFFFF0h set esp’s data the first and last bit to 0

Sub esp, 20h minus 20(in hex) from esp

Call \_\_\_main call the main

Mov dword ptr [esp+1Ch], 3 move 3 into dword ptr start at esp+1Ch

Mov dword ptr [esp+18h], 5 move 5 into dword ptr start at esp+18h

Mov dword ptr [esp+14h], 0 move 0 into dword ptr start at esp+14h

Mov eax, [esp+1Ch] move data in [esp+1Ch] to eax

Imul eax, [esp+18h] multiply data in [esp+18h] to eax

Mov edx, eax move eax’s data to edx

Mov ecx, [esp+1Ch] move data in [esp+1Ch] to ecx

Mov ecx, eax move eax’s data to ecx

shr ecx, 1Fh divide data in ecx by 2^31(set to 0)(no signal)

add eax, ecx add ecx’s data to eax

sar eax, 1 divide data in eax by 2 with signal

sub edx, eax minus eax’s value from edx

mov eax, edx move edx’s data to eax

mov [esp+14h], eax move eax’s data to [esp+14h]

mov eax, [esp+14h] move [esp+14h]’s value to eax

mov [esp+4], eax move eax’s data to [esp+4]

mov dword ptr [esp], offset aD ; "%d" move offset aD’s value to ptr[esp]

call printf call the print method

mov eax, 0 move 0 to eax

leave

retn

\_main endp

Q2:

.text:00401500 push ebp push ebp into stack

.text:00401501 mov ebp, esp move esp’s data into ebp

.text:00401503 and esp, 0FFFFFFF0h set the first 4 bit and last 4 bit in esp to 0

.text:00401506 sub esp, 40h minus 40h from esp

.text:00401509 call \_\_\_main call the main method

.text:0040150E mov dword ptr [esp+18h], 0Ch move 0Ch into dword list ptr at esp+18h

.text:00401516 mov dword ptr [esp+1Ch], 0Fh move 0Fh into dword list ptr at esp+1Ch

.text:0040151E mov dword ptr [esp+20h], 0DDh move 0DDh into dword list ptr at esp+20h

.text:00401526 mov dword ptr [esp+24h], 3 move 3 into dword list ptr at esp+24h

.text:0040152E mov dword ptr [esp+28h], 1B0h move 1B0h into dword list ptr at esp+28h

.text:00401536 mov dword ptr [esp+2Ch], 36h move 36h into dword list ptr at esp+2Ch

.text:0040153E mov dword ptr [esp+30h], 10h move 10h into dword list ptr at esp+30h

.text:00401546 mov dword ptr [esp+34h], 43h move 43h into dword list ptr at esp+34h

.text:0040154E mov dword ptr [esp+3Ch], 0 move 0 into dword list ptr at esp+3Ch

.text:00401556 mov dword ptr [esp+38h], 0 move 0 into dword list ptr at esp+38h

.text:0040155E jmp short loc\_40157F jump to location at 40157F

.text:00401560 ; ------------------------------------------------------

.text:00401560

.text:00401560 loc\_401560: ; CODE XREF: \_main+84↓j

.text:00401560 mov eax, [esp+38h] move data in [esp+38h] to eax  
.text:00401564 mov eax, [esp+eax\*4+18h] move data in [esp+eax\*4+18h] to eax

.text:00401568 cmp eax, [esp+3Ch] compare data between eax and [esp+3Ch]

(ZF=1 means the same, ZF=0 means different)

.text:0040156C jle short loc\_40157A

.text:0040156E mov eax, [esp+38h] move [esp+38h]’s data to eax

.text:00401572 mov eax, [esp+eax\*4+18h] move [esp+eax\*4+18h]’s data to eax

.text:00401576 mov [esp+3Ch], eax move eax’s data to [esp+3Ch]

.text:0040157A

.text:0040157A loc\_40157A: ; CODE XREF: \_main+6C↑j

.text:0040157A add dword ptr [esp+38h], 1 add 1 into dword list ptr at [esp+38h]

.text:0040157F jump to here

.text:0040157F loc\_40157F: ; CODE XREF: \_main+5E↑j

.text:0040157F cmp dword ptr [esp+38h], 7 compare 7 and the data in [esp+38h]

(ZF=1 means the same, ZF=0 means different)

.text:00401584 jle short loc\_401560

.text:00401586 mov eax, [esp+3Ch] move [esp+3Ch]’s data to eax

.text:0040158A mov [esp+4], eax move eax’s data into [esp+4]

.text:0040158E mov dword ptr [esp], offset aD ; "%d" move offset aD’s value to ptr[esp]

.text:00401595 call \_printf call the print method

.text:0040159A mov eax, 0 set eax to 0

.text:0040159F leave

.text:004015A0 retn

.text:004015A0 \_mainendp

Q3:

.text:00401500 push ebp push ebp into stack

.text:00401501 mov ebp, esp move esp’s data into ebp

.text:00401503 and esp, 0FFFFFFF0h set esp first and last 4 bit to 0

.text:00401506 sub esp, 20h minus 20h from esp

.text:00401509 call \_\_\_main call the main method

.text:0040150E mov dword ptr [esp+1Ch], 64h move 64h to dword list ptr at [esp+1Ch]

.text:00401516 jmp loc\_4015D6 jump to location 4015d6

.text:0040151B ; -----------------------------------------------------------------------

.text:0040151B

.text:0040151B loc\_40151B: ; CODE XREF: \_main+DE↓j

.text:0040151B mov ecx, [esp+1Ch] move [esp+1Ch]’s data to ecx

.text:0040151F mov edx, 51EB851Fh move 51EB851Fh(in hex) to edx

.text:00401524 mov eax, ecx move ecx’s data to eax

.text:00401526 imul edx multiply edx with sign

.text:00401528 sar edx, 5 divide data in eax by 2^5 with sign

.text:0040152B mov eax, ecx move ecx’s data to eax

.text:0040152D sar eax, 1Fh divide data in eax by 2^63 with signal

.text:00401530 sub edx, eax minus eax’s value from edx

.text:00401532 mov eax, edx move edx’s value to eax

.text:00401534 mov [esp+18h], eax move eax’s data to [esp+18h]

.text:00401538 mov eax, [esp+18h] move [esp+18h]’s data to eax

.text:0040153C imul edx, eax, -64h

.text:0040153F mov eax, [esp+1Ch] move [esp+1Ch]’s value to eax

.text:00401543 lea ecx, [edx+eax]

.text:00401546 mov edx, 66666667h move 66666667h to edx

.text:0040154B mov eax, ecx move ecx’s data to eax

.text:0040154D imul edx

.text:0040154F sar edx, 2 divide data in eax by 2^2 with sign

.text:00401552 mov eax, ecx move ecx’s data to eax

.text:00401554 sar eax, 1Fh divide data in eax by 2^63 with sign

.text:00401557 sub edx, eax minus eax’s data from edx

.text:00401559 mov eax, edx move edx’s data to eax

.text:0040155B mov [esp+14h], eax move eax’s data to [esp+14h]

.text:0040155F mov ecx, [esp+1Ch] move [esp+1Ch]’s data to ecx

.text:00401563 mov edx, 66666667h move 66666667h to edx

.text:00401568 mov eax, ecx move ecx’s data to eax

.text:0040156A imul edx

.text:0040156C sar edx, 2 divide data in eax by 2^2 with sign

.text:0040156F mov eax, ecx move ecx’s data to eax

.text:00401571 sar eax, 1Fh divide data in eax by 2^63 with sign

.text:00401574 sub edx, eax minus eax’s data from edx

.text:00401576 mov eax, edx move edx’s data to eax

.text:00401578 shl eax, 2 shift 2 bit to left on eax

.text:0040157B add eax, edx add the edx’s data on eax

.text:0040157D add eax, eax add the eax’s data on eax (2\*eax)

.text:0040157F sub ecx, eax minus eax’s data from ecx

.text:00401581 mov eax, ecx move ecx’s data to eax

.text:00401583 mov [esp+10h], eax move eax’s data to [esp+10h]

.text:00401587 mov eax, [esp+18h] move [esp+18h]’s data to eax

.text:0040158B imul eax, [esp+18h] multiply [esp+18h] data with eax’s data with sign

.text:00401590 imul eax, [esp+18h] multiply [esp+18h] data with eax’s data with sign

.text:00401595 mov edx, eax move eax’s data to eds

.text:00401597 mov eax, [esp+14h] move [esp+14h]’s data to eax

.text:0040159B imul eax, [esp+14h] multiply [esp+14h] data with eax’s data with sign

.text:004015A0 imul eax, [esp+14h] multiply [esp+14h] data with eax’s data with sign

.text:004015A5 add edx, eax add eax’s data to edx

.text:004015A7 mov eax, [esp+10h] move [esp+10h]’s data to eax

.text:004015AB imul eax, [esp+10h] multiply [esp+10h] data with eax’s data with sign

.text:004015B0 imul eax, [esp+10h] multiply [esp+10h] data with eax’s data with sign

.text:004015B5 add eax, edx add edx’s value to eax

.text:004015B7 cmp eax, [esp+1Ch] compare [esp+1Ch]’s data and eax’s data

(ZF=1 means the same, ZF=0 means different)

.text:004015BB jnz short loc\_4015D1

.text:004015BD mov eax, [esp+1Ch] move [esp+1Ch]’s data to eax

.text:004015C1 mov [esp+4], eax move eax’s data to [esp+4]

.text:004015C5 mov dword ptr [esp], offset aD ; "%d " move offset aD’s value to ptr[esp]

.text:004015CC call \_printf call the print method

.text:004015D1

.text:004015D1 loc\_4015D1: ; CODE XREF: \_main+BB↑j

.text:004015D1 add dword ptr [esp+1Ch], 1 add 1 to dword list ptr at [esp+1Ch]

.text:004015D6

.text:004015D6 loc\_4015D6: ; CODE XREF: \_main+16↑j

.text:004015D6 cmp dword ptr [esp+1Ch], 3E7h compare 3E7h to dword list ptr at [esp+1Ch]

.text:004015DE jle loc\_40151B

.text:004015E4 mov eax, 0 set eax to 0

.text:004015E9 leave

.text:004015EA retn

.text:004015EA \_mainendp