Get integer input from scanf

If input is lesser or equal to 0 then branch to other instructions, else compare input to see if it is greater than 10

```
    0x40061f <main+56>        call   0x4004f0 <__isoc99_scanf@plt>
    0x400624 <main+61>        mov    eax, DWORD PTR [rbp-0x4]
→   0x400627 <main+64>        test   eax, eax
    0x400629 <main+66>        jle    0x40067d <main+150>
```

Comparing input to 10(0xA)

```
0x40062e <main+71>        cmp    eax, 0xa
0x400631 <main+74>        jg     0x40067d <main+150>
```

This whole assembly code corresponds to

```
→   11      if(start>0 && start<11) {
```

Whole code consists of

1. Moving of variables, in this example, start is located at rbp-0x4
2. Loading argument into proper registers, rdi for message and format specifier, rsi for count value

```
gef➤  x/s 0x400752
0x400752:       "Countdown -> %d\n"
gef➤
```

```
gef➤  x/wx $rbp-4
0x7fffffffe41c: 0x00000004
gef➤
```

```
    0x40063d <main+86>        mov    eax, DWORD PTR [rbp-0x4]
→   0x400640 <main+89>        mov    esi, eax
    0x400642 <main+91>        lea    rdi, [rip+0x109]        # 0x400752
    0x400649 <main+98>        mov    eax, 0x0
    0x40064e <main+103>       call   0x4004e0 <printf@plt>
```

Whole code consists of

1. Loading variable value from memory to register and decrementing it
2. Does a comparison and if value is greater than zero, loop continues

```
     0x400653 <main+108>       mov    eax, DWORD PTR [rbp-0x4]
 →   0x400656 <main+111>       sub    eax, 0x1
     0x400659 <main+114>       mov    DWORD PTR [rbp-0x4], eax
     0x40065c <main+117>       mov    eax, DWORD PTR [rbp-0x4]
     0x40065f <main+120>       test   eax, eax
     0x400661 <main+122>       jg     0x40063d <main+86>
```

```
gef➤  print $eflags
$1 = [ PF IF ]
gef➤
```

If start is 0 or lesser

```
→   0x400661 <main+122>      jg     0x40063d <main+86>       NOT taken [Reason: !(!Z && S==O)]
```

```
gef➤  print $eflags
$2 = [ PF ZF IF ]
```

This whole code consists of printing messages after the loop

```
   0x400663 <main+124>       lea    rdi, [rip+0xf9]        # 0x400763
   0x40066a <main+131>       call   0x4004d0 <puts@plt>
   0x40066f <main+136>       lea    rdi, [rip+0xf5]        # 0x40076b
   0x400676 <main+143>       call   0x4004d0 <puts@plt>
```

Return(0)

```
 →   0x400689 <main+162>       mov    eax, 0x0
     0x40068e <main+167>       leave
     0x40068f <main+168>       ret
```