CS 231 - DLDCA Lab

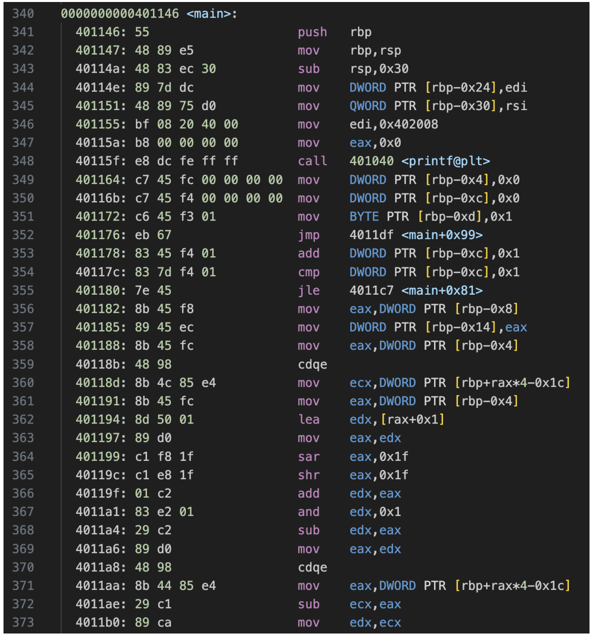
Lab Assignment 3

Report

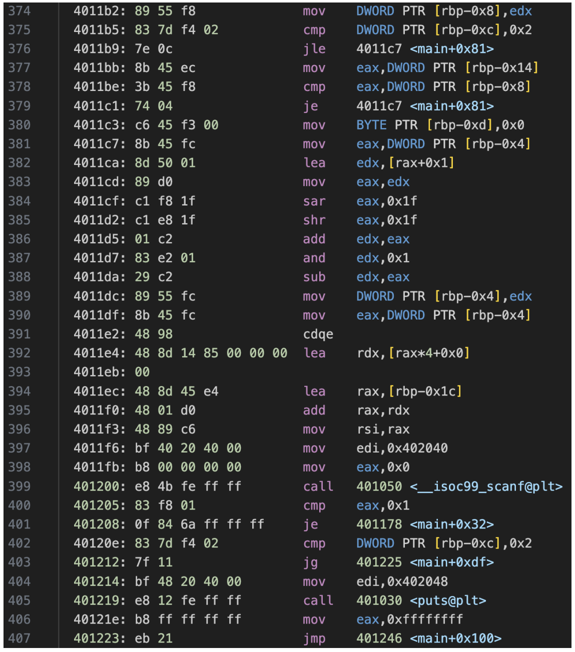
Aditya Singh - 22B1844

Program 1

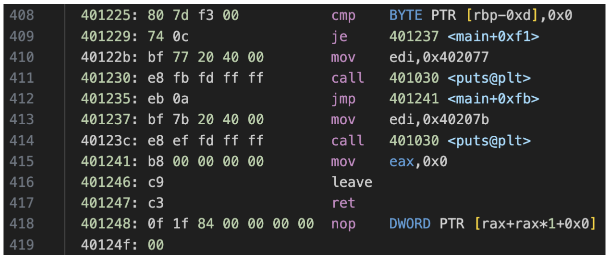
This program asks the user to enter ≤ 3 numbers and if they form an AP then it prints YES and otherwise it prints NO.
This can be determined by analysing the mai function of the assembly code of the program which has been attached below.



```
0000000000401146 <main>:
  401146: 55                      push   rbp
  401147: 48 89 e5                mov    rbp,rsp
  40114a: 48 83 ec 30             sub    rsp,0x30
  40114e: 89 7d dc                mov    DWORD PTR [rbp-0x24],edi
  401151: 48 89 75 d0             mov    QWORD PTR [rbp-0x30],rsi
  401155: bf 08 20 40 00          mov    edi,0x402008
  40115a: b8 00 00 00 00          mov    eax,0x0
  40115f: e8 dc fe ff ff          call   401040 <printf@plt>
  401164: c7 45 fc 00 00 00 00    mov    DWORD PTR [rbp-0x4],0x0
  40116b: c7 45 f4 00 00 00 00    mov    DWORD PTR [rbp-0xc],0x0
  401172: c6 45 f3 01             mov    BYTE PTR [rbp-0xd],0x1
  401176: eb 67                   jmp    4011df <main+0x99>
  401178: 83 45 f4 01             add    DWORD PTR [rbp-0xc],0x1
  40117c: 83 7d f4 01             cmp    DWORD PTR [rbp-0xc],0x1
  401180: 7e 45                   jle    4011c7 <main+0x81>
  401182: 8b 45 f8                mov    eax,DWORD PTR [rbp-0x8]
  401185: 89 45 ec                mov    DWORD PTR [rbp-0x14],eax
  401188: 8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
  40118b: 48 98                   cdqe
  40118d: 8b 4c 85 e4             mov    ecx,DWORD PTR [rbp+rax*4-0x1c]
  401191: 8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
  401194: 8d 50 01                lea    edx,[rax+0x1]
  401197: 89 d0                   mov    eax,edx
  401199: c1 f8 1f                sar    eax,0x1f
  40119c: c1 e8 1f                shr    eax,0x1f
  40119f: 01 c2                   add    edx,eax
  4011a1: 83 e2 01                and    edx,0x1
  4011a4: 29 c2                   sub    edx,eax
  4011a6: 89 d0                   mov    eax,edx
  4011a8: 48 98                   cdqe
  4011aa: 8b 44 85 e4             mov    eax,DWORD PTR [rbp+rax*4-0x1c]
  4011ae: 29 c1                   sub    ecx,eax
  4011b0: 89 ca                   mov    edx,ecx
```



```
  4011b2: 89 55 f8                mov    DWORD PTR [rbp-0x8],edx
  4011b5: 83 7d f4 02             cmp    DWORD PTR [rbp-0xc],0x2
  4011b9: 7e 0c                   jle    4011c7 <main+0x81>
  4011bb: 8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
  4011be: 3b 45 f8                cmp    eax,DWORD PTR [rbp-0x8]
  4011c1: 74 04                   je     4011c7 <main+0x81>
  4011c3: c6 45 f3 00             mov    BYTE PTR [rbp-0xd],0x0
  4011c7: 8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
  4011ca: 8d 50 01                lea    edx,[rax+0x1]
  4011cd: 89 d0                   mov    eax,edx
  4011cf: c1 f8 1f                sar    eax,0x1f
  4011d2: c1 e8 1f                shr    eax,0x1f
  4011d5: 01 c2                   add    edx,eax
  4011d7: 83 e2 01                and    edx,0x1
  4011da: 29 c2                   sub    edx,eax
  4011dc: 89 55 fc                mov    DWORD PTR [rbp-0x4],edx
  4011df: 8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
  4011e2: 48 98                   cdqe
  4011e4: 48 8d 14 85 00 00 00    lea    rdx,[rax*4+0x0]
  4011eb: 00
  4011ec: 48 8d 45 e4             lea    rax,[rbp-0x1c]
  4011f0: 48 01 d0                add    rax,rdx
  4011f3: 48 89 c6                mov    rsi,rax
  4011f6: bf 40 20 40 00          mov    edi,0x402040
  4011fb: b8 00 00 00 00          mov    eax,0x0
  401200: e8 4b fe ff ff          call   401050 <__isoc99_scanf@plt>
  401205: 83 f8 01                cmp    eax,0x1
  401208: 0f 84 6a ff ff ff       je     401178 <main+0x32>
  40120e: 83 7d f4 02             cmp    DWORD PTR [rbp-0xc],0x2
  401212: 7f 11                   jg     401225 <main+0xdf>
  401214: bf 48 20 40 00          mov    edi,0x402048
  401219: e8 12 fe ff ff          call   401030 <puts@plt>
  40121e: b8 ff ff ff ff          mov    eax,0xffffffff
  401223: eb 21                   jmp    401246 <main+0x100>
```



```
408    401225: 80 7d f3 00           cmp    BYTE PTR [rbp-0xd],0x0
409    401229: 74 0c                 je     401237 <main+0xf1>
410    40122b: bf 77 20 40 00        mov    edi,0x402077
411    401230: e8 fb fd ff ff        call   401030 <puts@plt>
412    401235: eb 0a                 jmp    401241 <main+0xfb>
413    401237: bf 7b 20 40 00        mov    edi,0x40207b
414    40123c: e8 ef fd ff ff        call   401030 <puts@plt>
415    401241: b8 00 00 00 00        mov    eax,0x0
416    401246: c9                    leave
417    401247: c3                    ret
418    401248: 0f 1f 84 00 00 00 00  nop    DWORD PTR [rax+rax*1+0x0]
419    40124f: 00
```

Lines 341 to 348 prompt the user to input numbers. In lines 349,350 two variables of the type DWORD which are rbp-0x4 and rbp -0xc, are initialised to zero, and on 351, a variable of the type BYTE is initialised to 1. BYTE type here implies a variable of type bool because later we see that at line 408 it is compared to 0.

Now we jump to line 390 where the rbp-0x4 is stored into eax which is then expanded using cdqe from 32 bits to 64 bits so that it is now called rax.

Now the value 4 times rax is stored into rdx. This gives us an idea that rdx is storing the address of something to which rax points. This is because we multiply rax by 4, and addresses are multiples of 4. So we have an idea that rbp-0x4 is storing the index to elements of an array of integers and rdx is pointing to the address of the elements of that array.

Now at line 399, the program prompts the user to give input. After taking the first input eax stores 1 and we jump to line 353 where rbp-0xc is increased by 1, so it appears that rbp-0xc is used to store the number of inputs been given by the user. Then at line 354 it is checked if rbp-0xc is ≤ 2 if it is then we jump to 381. At line 382 value of 1 + rax is stored into edx. 1 + rax is basically rbp-0x4 + 1. At line 384, the bits of eax are shifted forward by 31 signed bits using the sar function. Then at line 385 shr reduces eax to 0 because rbp-0x4 cannot be negative. Then at line 389 we store the value of edx into rbp-0x4. edx was the opposite of the original value of rbp-0x4. That is if original value of rbp-0x4 was 1 then edx was 0 and vice versa. So in line 389, the value of rbp-0x4 is reversed. rbp-0x4 is also used an array index, by these facts we determine that the numbers recieved from the user are being stored in an array which has size 2.

Now we are again at line 391 and the program goes on.

Now we consider the case in which the condition of line 354 is not satisfied. Here two new variables are declared rbp-0x8 and rbp-0x14. Then we subtract both of them and store the result in rbp-0x14. The next lines again do the reversal of rbp-0x4. eax stores the value of rbp-0x4 and edx stores the reverse of the value of rbp-0x4. Next we subtract the values of edx and eax and store this in rbp-0x8. Basically, rbp-0x14 contains the difference of second last pair of inputs and rbp-0x8 contains the difference of last pair of inputs. At line 378 we compare the value of rbp-0x14 and rbp-0x8, if they are equal we move forward and otherwise we set rbp-0xd to 0.

When the user enters the last input the value of rbp-0xd is checked, if it is zero then NO gets printed and otherwise YES is printed.

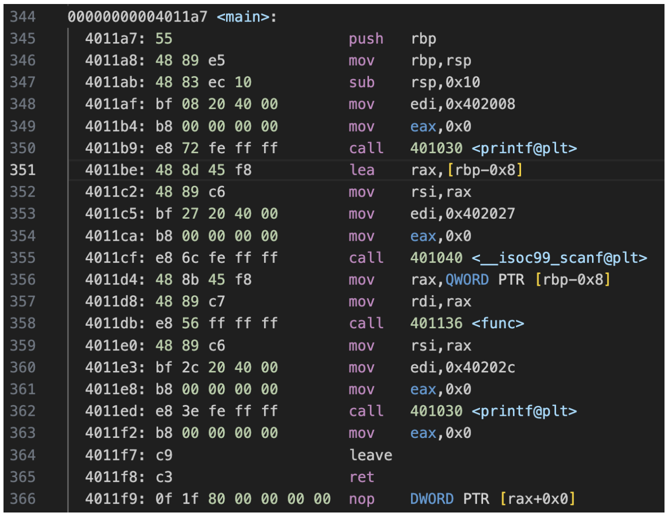
Conclusion:

As we are comparing the difference of pairs of consecutive inputs to see f they are equal we are essentially checking the condition of AP. Thus the program tests whether the input sequence is an AP or not.

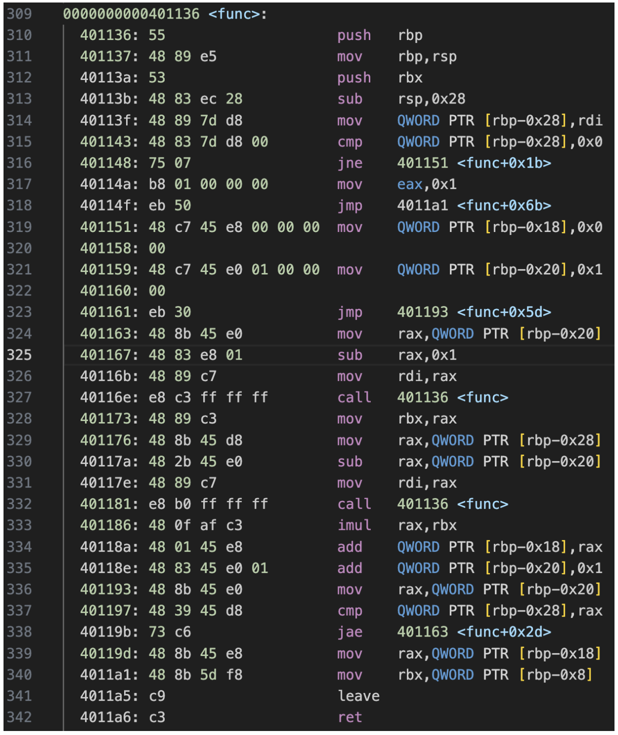
Program 2

The code generates the Catalan numbers. We can get to know this from the main function and the function func which does the major computation of the assembly code.

The main function and the other function func:



```
00000000004011a7 <main>:
  4011a7: 55                     push   rbp
  4011a8: 48 89 e5               mov    rbp,rsp
  4011ab: 48 83 ec 10            sub    rsp,0x10
  4011af: bf 08 20 40 00         mov    edi,0x402008
  4011b4: b8 00 00 00 00         mov    eax,0x0
  4011b9: e8 72 fe ff ff         call   401030 <printf@plt>
  4011be: 48 8d 45 f8            lea    rax,[rbp-0x8]
  4011c2: 48 89 c6               mov    rsi,rax
  4011c5: bf 27 20 40 00         mov    edi,0x402027
  4011ca: b8 00 00 00 00         mov    eax,0x0
  4011cf: e8 6c fe ff ff         call   401040 <__isoc99_scanf@plt>
  4011d4: 48 8b 45 f8            mov    rax,QWORD PTR [rbp-0x8]
  4011d8: 48 89 c7               mov    rdi,rax
  4011db: e8 56 ff ff ff         call   401136 <func>
  4011e0: 48 89 c6               mov    rsi,rax
  4011e3: bf 2c 20 40 00         mov    edi,0x40202c
  4011e8: b8 00 00 00 00         mov    eax,0x0
  4011ed: e8 3e fe ff ff         call   401030 <printf@plt>
  4011f2: b8 00 00 00 00         mov    eax,0x0
  4011f7: c9                     leave
  4011f8: c3                     ret
  4011f9: 0f 1f 80 00 00 00 00   nop    DWORD PTR [rax+0x0]
```



```
0000000000401136 <func>:
  401136: 55                    push   rbp
  401137: 48 89 e5              mov    rbp,rsp
  40113a: 53                    push   rbx
  40113b: 48 83 ec 28           sub    rsp,0x28
  40113f: 48 89 7d d8           mov    QWORD PTR [rbp-0x28],rdi
  401143: 48 83 7d d8 00        cmp    QWORD PTR [rbp-0x28],0x0
  401148: 75 07                 jne    401151 <func+0x1b>
  40114a: b8 01 00 00 00        mov    eax,0x1
  40114f: eb 50                 jmp    4011a1 <func+0x6b>
  401151: 48 c7 45 e8 00 00 00  mov    QWORD PTR [rbp-0x18],0x0
  401158: 00
  401159: 48 c7 45 e0 01 00 00  mov    QWORD PTR [rbp-0x20],0x1
  401160: 00
  401161: eb 30                 jmp    401193 <func+0x5d>
  401163: 48 8b 45 e0           mov    rax,QWORD PTR [rbp-0x20]
  401167: 48 83 e8 01           sub    rax,0x1
  40116b: 48 89 c7              mov    rdi,rax
  40116e: e8 c3 ff ff ff        call   401136 <func>
  401173: 48 89 c3              mov    rbx,rax
  401176: 48 8b 45 d8           mov    rax,QWORD PTR [rbp-0x28]
  40117a: 48 2b 45 e0           sub    rax,QWORD PTR [rbp-0x20]
  40117e: 48 89 c7              mov    rdi,rax
  401181: e8 b0 ff ff ff        call   401136 <func>
  401186: 48 0f af c3           imul   rax,rbx
  40118a: 48 01 45 e8           add    QWORD PTR [rbp-0x18],rax
  40118e: 48 83 45 e0 01        add    QWORD PTR [rbp-0x20],0x1
  401193: 48 8b 45 e0           mov    rax,QWORD PTR [rbp-0x20]
  401197: 48 39 45 d8           cmp    QWORD PTR [rbp-0x28],rax
  40119b: 73 c6                 jae    401163 <func+0x2d>
  40119d: 48 8b 45 e8           mov    rax,QWORD PTR [rbp-0x18]
  4011a1: 48 8b 5d f8           mov    rbx,QWORD PTR [rbp-0x8]
  4011a5: c9                    leave
  4011a6: c3                    ret
```

From the code of the main function it is clear that the lines 345 to 355 are used to prompt the user to input a number and receive the input which is then stored in rdi register in the lines 356 and 357.

In the line 358, the function func is called which calculates the output for the user input.

Let's see what does func do:

Initially the value of the input which was stored in rdi is copied to rbp-0x28. Then this value is compared to 0, and if it is zero then the register eax is assigned the value 1 and the function returns. This is the base case of the recursion.

If the input is not 0 then we jump to the line 319 and values 0 and 1 are assigned to rbp-0x18 and rbp-0x20 respectively. At line 323 there is a jump statement without any conditions which means that there is a loop. Then the value of (rbp-0x20) - 1 is stored into rdi and the function is called again with this value of rdi as input. The output of this call is stored in rax, which is then copied to rbx. Then at line 332 the function is again called with input as (rbp-0x28) - (rbp-0x20) which is basically 1 less than the previous input. Now at line 333 the value of the product (Output for current input)*(Output for previous input) is stored in rax. At line 334 this value of rax is added to the register rbp-0x18 which had been initialised to zero in line 319. **The main observation is that in each iteration of the loop the above product is calculated and is added to rbp-0x18, which is later used as the final output as seen below.**

After this the value of rbp-0x20 is increased by 1 which is then stored in rax. This value of rax is then compared with rbp-0x28 which was the original input, if rax is lesser than the original input then we exit the loop otherwise we again go in the loop.

When we finally exit the loop the value stored in rbp-0x18 is copied into rax which is taken as the final output.

From this analysis we have determined the recursive relation for the function func as:

$$func(0) = 1$$

$$func(n) = \sum_{i=1}^{n} func(n-i)func(i-1), n \neq 0$$

This recursion defines a very famous sequence of numbers called the Catalan Numbers which have a very wide application in mathematics.