DLDCA Lab3 Q1 Report

Jaladhi Joshi : 22B0994

6th October, 2023

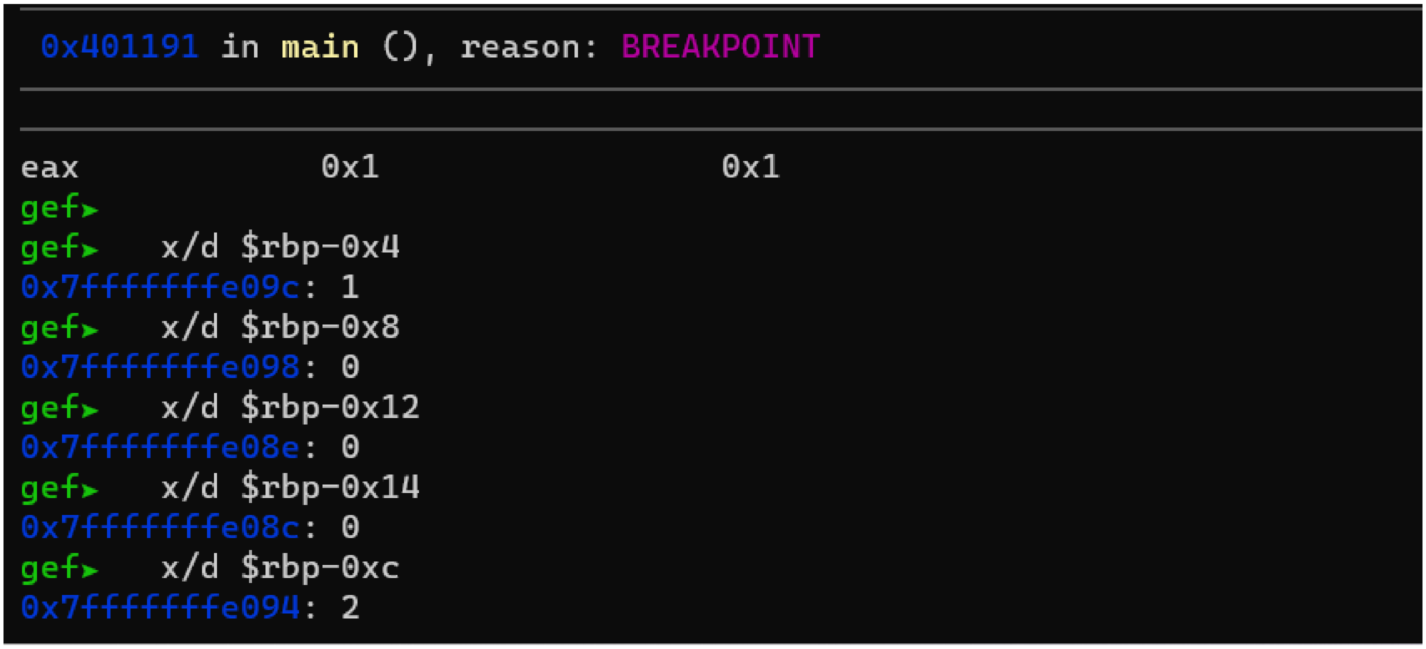
0.1 Lab-3 Q1

0.1.1 Program1

The sequences that are accepted are **Arithmetic Progressions**. This is concluded by examining the code as follows:

At first I used the command : **objdump -d program1 -M intel**

I analysed the **main** function of the disassembly code which is from memory address 0x401146 to 0x40124f using gef.
I printed out the values of registers $rbp-0x4, $rbp-0x8, $rbp-0x12, $rbp-0x14 and eax .This gave me a rough idea of what is happening in the code.



```
 0x401191 in main (), reason: BREAKPOINT

eax            0x1                 0x1
gef➤
gef➤  x/d $rbp-0x4
0x7fffffffe09c: 1
gef➤  x/d $rbp-0x8
0x7fffffffe098: 0
gef➤  x/d $rbp-0x12
0x7fffffffe08e: 0
gef➤  x/d $rbp-0x14
0x7fffffffe08c: 0
gef➤  x/d $rbp-0xc
0x7fffffffe094: 2
```

We can divide the entire code of the main function into small subgroups as follows:

- 0x401146-0x401176
- 0x401178-0x40117c
- 0x401182-0x4011b5
- 0x4011bb-0x4011be
- 0x4011c3-0x401205
- 0x40120e
- 0x401214-0x40124f

⇒ The code starts from the code snippet 0x401146-0x401176.

```
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>
```

This part of the code initialises the variables and prints ”Enter three or more numbers (Terminate with CTRL + D): ” and then jumps unconditionally to 0x4011df

```
  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>
```

⇒Here,rax*4+0x0 represents that from this value we can keep getting values that we have input,rax represents an index and rax*4+0x0 can iterate through that array.

⇒ Scanf is called , which takes in the input, eax represents whether input is there or not

⇒In case of no input eax at this place would be 0 and the comparison will yeild not equal to,hence if eax=1, then the program moves to 0x401178 or else if moves to the address 0x40120e.

⇒ Let us first take the case where eax=0 and we move to 0x40120e

```
  40120e:      83 7d f4 02              cmp    DWORD PTR [rbp-0xc],0x2
  401212:      7f 11                    jg     401225 <main+0xdf>
```

⇒ Here a comparison is made between $rbp-0xc and 2,$rbp-0xc stores the number of elements,if the number of elements are > 2 then we move to 0x401225 else we move to 0x401214.

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

⇒ If the number of elements are > 2,then we go to 0x401225 and if the value of $rbp-0xd is 0, we print ”No”,else we print ”Yes”

```
  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
```

⇒ If the number of elements are <= 2, then we print ”You have not entered enough numbers”

⇒ Now the case when eax=1,input is to be processed .

While processing the input it first checks if the number of inputs already present,$rbp-0xc is greater than one or not,if not one,it takes more inputs or exits,when the number of inputs is greater than one,than we start the actual sequence identifying loop.

```
  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
```

⇒ As we know that we can get elements of the array by toggling with index $rbp+rax*4-0x1c , we use that array for comparisons

⇒Initially $rbp+rax*4-0x1c is stored in $ecx, $rbp-0x4 is stored in eax,which is either 0 or 1 and tells us about the latest element added, the sar ,shr code block essentially makes eax 1 if it was 0 and vice-versa

⇒Initial element stored in ecx was the latest input and after changing eax, now ecx contains the input just before the latest input and their difference is stored in $rbp-0x8.

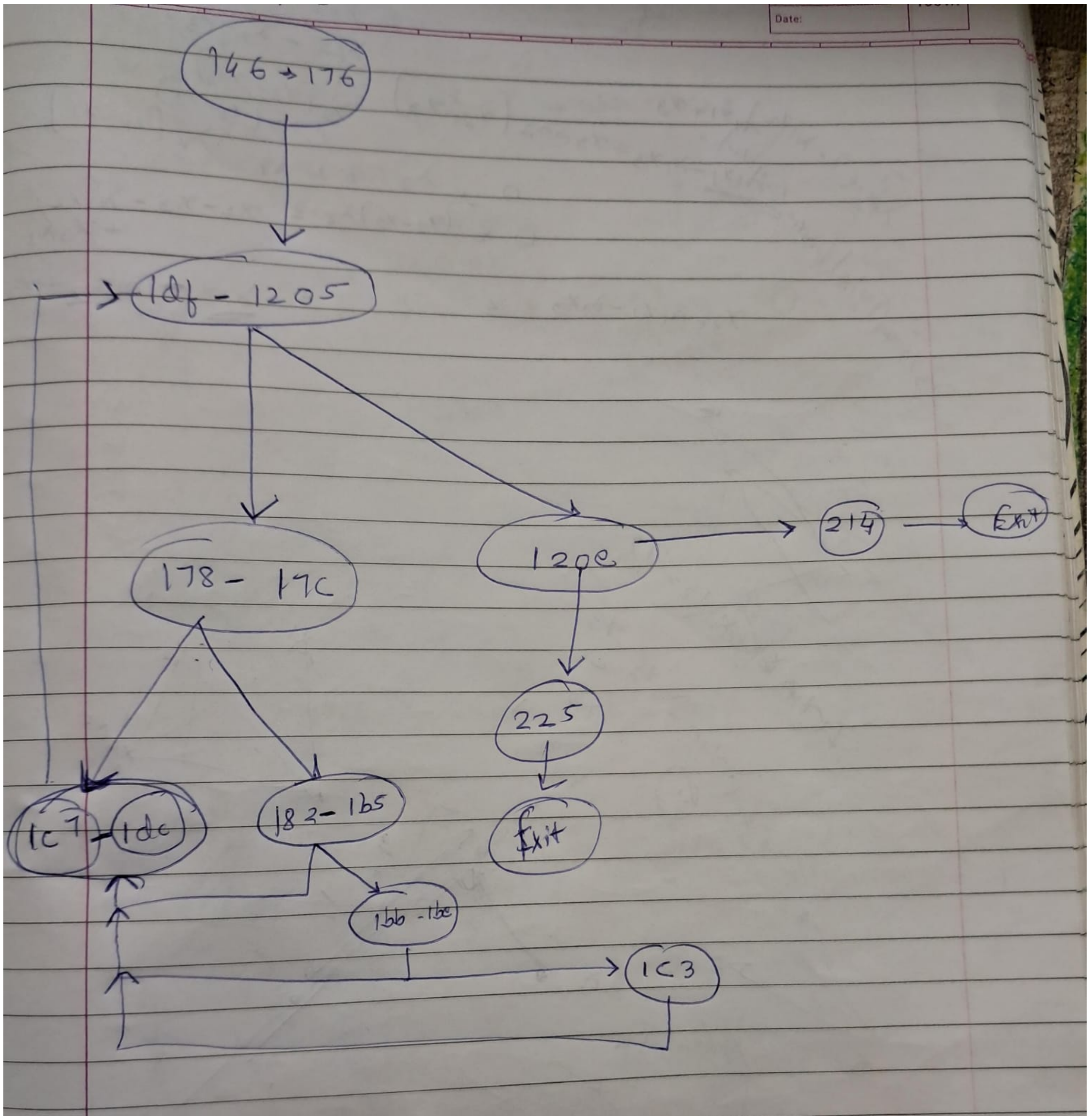
⇒ Again the number of elements greater than 2 condition in checked and if true we proceed further

```
  4011bb:         8b 45 ec                    mov     eax,DWORD PTR [rbp-0x14]
  4011be:         3b 45 f8                    cmp     eax,DWORD PTR [rbp-0x8]
```

⇒ $rbp-0x8 has the current difference of consecutive elements and $rbp-0x14 has the previous difference,if they are equal we keep checking,if not we assign $rbp-0xd the value 0 ,output is yes when this value is 1 which is only when sequence is AP. The registers represent:

- $rbp-0x4-Tells which input is the latest input
- $rbp-0x24 and $rbp-0x28-One of these has the latest input
- $rbp-0x8- Current difference of consecutive elements
- $rbp-0x14- Previous difference of consecutive elements
- $rbp-0xc-Number of elements
- $rbp-0xd-Decides output

Overall,the flow of the program is as follows:



0.1.2 Program2

Main:

```
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

011db:      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
```

The main function first takes the input, calls the function,the function returns the value in rax which is moved in rsi and that value gets printed along with the output string.

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>
```

⇒The input is stored in rdi and hence in $rbp-0x28 here and variable initializations are done as well as base case,that if input is 0 push 1 in rax

```
401151:     48 c7 45 e8 00 00 00     mov    QWORD PTR [rbp-0x18],0x0
401159:     48 c7 45 e0 01 00 00     mov    QWORD PTR [rbp-0x20],0x1
```

⇒ If input is not 0, we initialize two more variables

```
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
```

⇒ Here one function call is made for $rbp-0x20 -1 ,then that input is stored in rbx

⇒Another function call for input - $rbp-0x20

⇒Both the values returned are multiplied and added to the initial value of $rbp-0x18

⇒The value of $rbp-0x20 is increased till its less than the input

⇒$rbp-0x18 has the desired output which is moved to rax

⇒ Hence,

$$f(n) = \sum_{i=1}^{n} f(i)f(n-i)$$

⇒

$$f(n) = \frac{\binom{2n}{n}}{n+1}$$