# CS232 : Lab3 - Part1

Thomas Biju Cheeramvelil (22B1073)

## Question 1

### Program 1

#### Result

If the number of inputs entered is less than 3, the program outputs "Insufficient number of inputs". Else, the program outputs YES, if the inputs entered form an arithmetic progression of integers (4 bytes, $-2^{31}$ to $2^{31}-1$), and NO if not.

#### Approach

The command `objdump -D -Mintel program1` is first run on the terminal to view the assembly file corresponding to the given object file `program1`.

Program1: main

```
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>
  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
```

### Memory locations used in <main>

1. [rbp-0x1c] and [rbp-0x18] are used for storing the current input and the previous input, and are used for finding the difference between them. The inputs are toggled between them, that is, the first input is stored in [rbp-0x1c], the second in [rbp-0x18], and so on alternatively.

2. [rbp-0x4] contains the toggling value, which is used for storing the inputs alternatively in the above memory locations.

3. [rbp-0x8] contains the recent difference between inputs and [rbp-0x14] contains the previous difference between inputs.

4. [rbp-0xc] contains the number of inputs entered until now. It was initialised as 0 and incremented by 1 at the beginning of the loop, at line 401178.

5. [rbp-0xd] contains a boolean value (1 byte) which indicates whether the inputs entered until now are in arithmetic progression or not. It was initialised as 1, and is changed to 0 if the any input is not in the same arithmetic progression as the previous inputs.

### Observations

1. The beginning lines from 401146 to 401151 pushes the base pointer `rbp` into the stack and makes the stack pointer `rsp` the new base pointer, thereby allocating the memory for the variables. The lines 401155 to 40115f calls the function `<printf@plt>` which prints the input message "Enter three or more numbers (Terminate with CTRL + D):". Then, the lines 401164 to 401176 initializes the 4-byte integer variables `[rbp-0x4] and [rbp-0xc]`, which store the toggling value and the number of inputs respectively, to both 0, and the 1-byte boolean variable `[rbp-0xd]`, which stores the flag, to 1. Then, the control jumps to 4011df.

2. The main loop of the program is from line 401178 to 401200. The input is taken from the lines 4011f3 to 401200, by calling the function `<__isoc99_scanf@plt>`. This function stores the inputted value in the `rsi` register, similar to the second argument &i in the scanf("%d",&i) which stores the input. The `rsi` register was set to the memory location `[rbp-0x1c]` or `[rbp-0x18]`, depending on the number of inputs already received. If there is an input, `eax` becomes the number of inputs which is 1, whereas if there is no input, i.e., `eax` becomes 0 and exits the loop.

3. Inside the loop, at line 4011be, the previous common difference and the current common difference are compared, and the flag `[rbp-0xd]` is set to 0 if they are not equal.

4. When the loop is exited, if the number of inputs is less than 3, "Insufficient number of inputs" is printed in the lines from 401214 to 401223 and exits main. If it is greater than or equal to 3, then depending on whether the flag stored at `[rbp-0xd]` is 1 or 0, "YES" or "NO" are printed respectively.

5. Thus, the program stores the current input, previous input, previous common difference, computes the current difference between the current input and previous input, and checks whether the current and previous difference are equal, and accepts the sequence only if the difference is maintained throughout the sequence. Hence, this program accepts arithmetic progressions of integers.

## Program 2

### Result

By reading the x86-64 assembly code, found by disassembling the binary executable file, it can be shown that the recurrence relation the function $func(n)$ satisfies is

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

and that this recurrence is satisfied by $func(n) = n$-th Catalan number $= \dfrac{^{2n}C_n}{n+1}$.

### Approach

The command `objdump -D -Mintel program2` is first run on the terminal to view the assembly file corresponding to the given object file `program2`.

Program2: func and main

```
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

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]
```

### Memory locations used in <func>

1. [rbp-0x18] contains the partial sum of the sum in the recurrence relation until the current iteration.
2. [rbp-0x20] contains the iteration index $i$. It is initialized as 1 at line 401159, and incremented by 1 at line 40118e.
3. [rbp-0x28] contains the argument $n$ to the current function call, which is assigned the value of register rdi at line 40113f.

### Registers used in <func>

1. rbp - (base pointer) points to the base address of the current stack frame.
2. rsp - (stack pointer) used to keep track of the current position of the stack.
3. rdi - (destination index) often used for passing the first argument to a function in x86-64. Can also used as a general-purpose register.
4. rsi - (source index) typically used for passing the second argument to a function in x86-64. Can also used as a general-purpose register.
5. rax, rbx - General purpose registers
6. eax, edi - Lower 32-bit halves of rax, rdi

### Observations

1. The beginning lines from 401146 to 401151 pushes the base pointer `rbp` into the stack and makes the stack pointer `rsp` the new base pointer, thereby allocating the memory for the variables. The lines 401155 to 40115f calls the function `<printf@plt>` which prints the input message "Enter a non-negative integer: ". The input is read and stored in `[rbp-0x8]` and `rax`.
2. Inside the function, there is a loop from lines 401163 and 40119b. The iteration index is stored in `[rbp-0x20]` contains the iteration index $i$. From lines 401163 to 40116e, `rdi` is set as $i-1$ which acts as the argument for the function, and the result `func(i-1)` is stored in `rbx`. Similarly from lines 401173 to 401181, `func(n-i)` is stored in `rax`.
3. Then, the product of `func(i-1)` and `func(n-i)` is calculated and added to the partial sum stored in `[rbp-0x18]`. The loop is exited when the iteration index $i$ becomes $n$, the input to the program.
4. Hence, we get the required recurrence relation

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

.