https://commons.wikimedia.org/w/index.php?curid=274746

Mathematical Foundations

More Recursion

CS51 – Spring 2026

Revisiting rec_sum

- Write a recursive function called rec_sum that takes a positive number as a parameter and calculates the sum of the numbers from 0 up to and including that provided number.

```
def rec_sum(n):
    if n == 0:
        return 0
    else:
        return n + rec_sum(n-1)
```

rec_sum function in cs51 machine

```
    lcw r1 stack              ; set up stack
    loa r3 r0                 ; get n and store it in r3

    lcw r2 rec_sum            ; call rec_sum
    cal r2 r2

    sto r3 r0                 ; print result,
    hlt                       ; and halt
```

…

```
    dat 100
stack
```

rec_sum function in cs51 machine

```
rec_sum
        psh r2   ; save the return address on the stack
        bne r3 r0 recurse   ; if n!=0 recurse
        add r3 r0 0         ; if n == 0, result is 0
        brs done

recurse
        psh r3              ; save n on the stack
        sub r3 r3 1         ; n = n-1

        lcw r2 rec_sum      ; make recursive call
        cal r2 r2 ; rec_sum (n-1), answer should be in r3

        pop r2              ; get n into r2
        add r3 r3 r2        ; r3 = n + rec_sum (n-1)
done
        pop r2              ; get the return address
        jmp r2              ; go back to caller
```

Function startup

base case

recursive case

Function cleanup and return

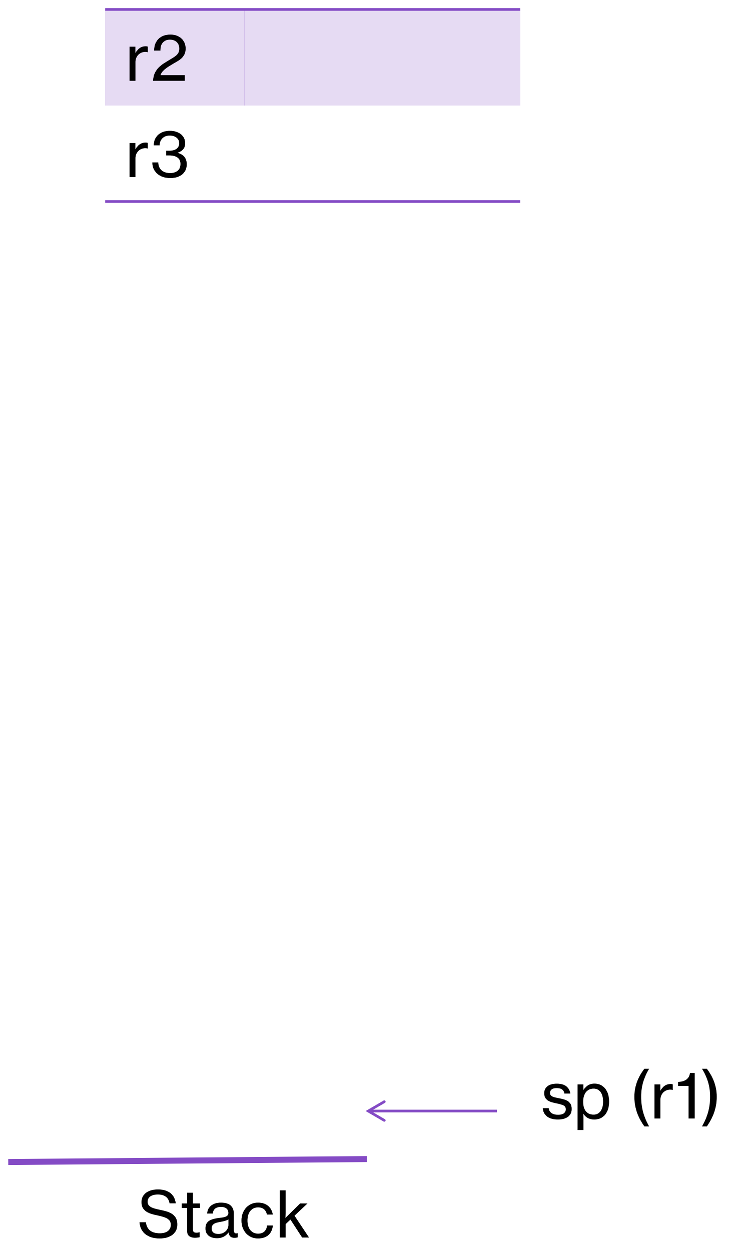
rec_sum function in cs51 machine

```
rec_sum
        psh r2    ; save the return address on the stack
        bne r3 r0 recurse    ; if n!=0 recurse
        add r3 r0 0          ; if n == 0, result is 0
        brs done

recurse
        psh r3               ; save n on the stack
        sub r3 r3 1          ; n = n-1

        lcw r2 rec_sum       ; make recursive call
        cal r2 r2 ; rec_sum (n-1), answer should be in r3

        pop r2               ; get n into r2
        add r3 r3 r2         ; r3 = n + rec_sum (n-1)
done
        pop r2               ; get the return address
        jmp r2               ; go back to caller
```

Notice the symmetry between push and pop

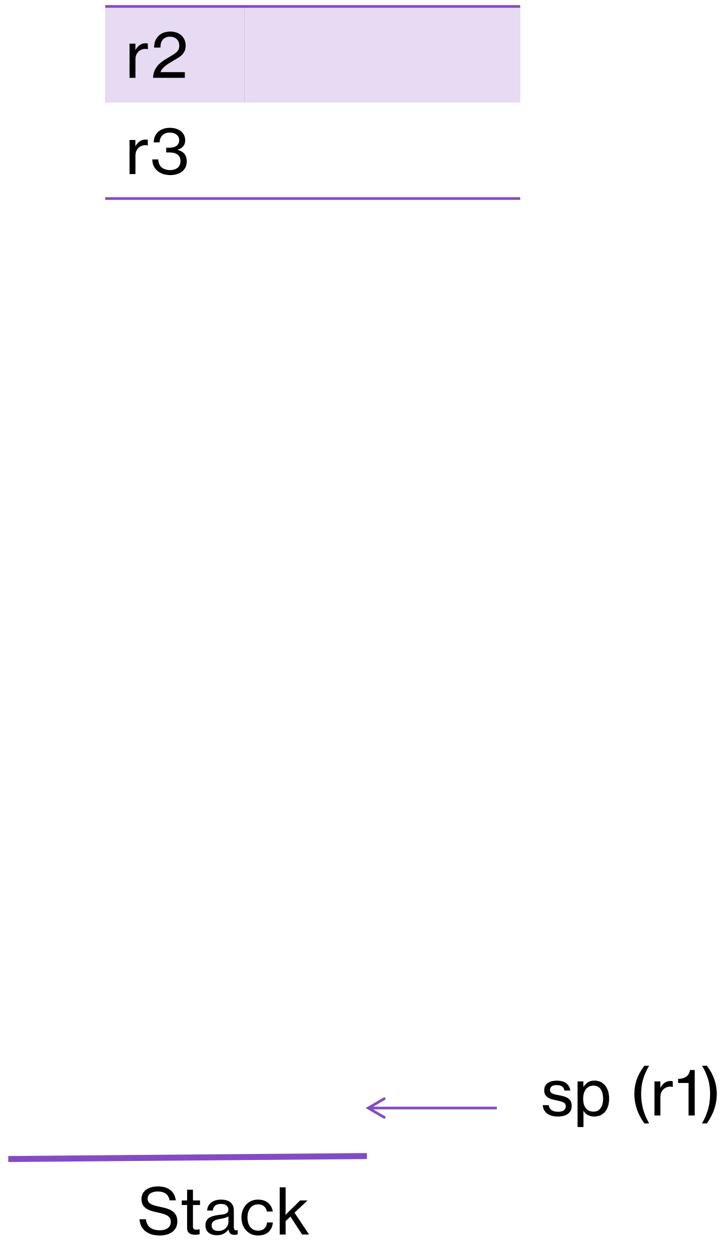
Calling `rec_sum`

```
2         lcw r1 stack
6         loa r3 r0

8         lcw r2 rec_sum
12        cal r2 r2

14        sto r3 r0
16        hlt


rec_sum
18        psh r2
22        bne r3 r0 recurse
24        add r3 r0 0
26        brs done

...
```



Calling `rec_sum`

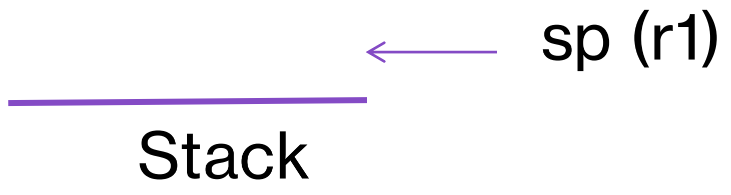
```
2         lcw r1 stack
6         loa r3 r0

8         lcw r2 rec_sum
12        cal r2 r2

14        sto r3 r0
16        hlt


rec_sum
18        psh r2
22        bne r3 r0 recurse
24        add r3 r0 0
26        brs done

...
```



Calling `rec_sum`

```
2         lcw r1 stack
6         loa r3 r0

8         lcw r2 rec_sum
12        cal r2 r2

14        sto r3 r0
16        hlt


rec_sum
18        psh r2
22        bne r3 r0 recurse
24        add r3 r0 0
26        brs done

...
```

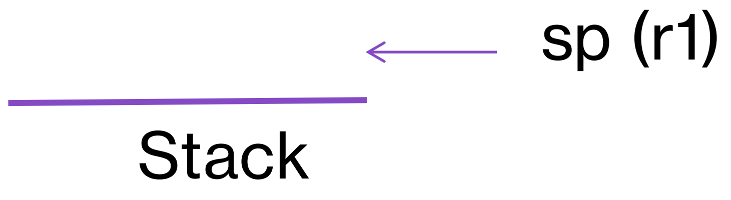
r2	
r3	2



Calling `rec_sum`

```
2        lcw r1 stack
6        loa r3 r0

8        lcw r2 rec_sum
12       cal r2 r2

14       sto r3 r0
16       hlt



rec_sum
18       psh r2
22       bne r3 r0 recurse
24       add r3 r0 0
26       brs done

...
```

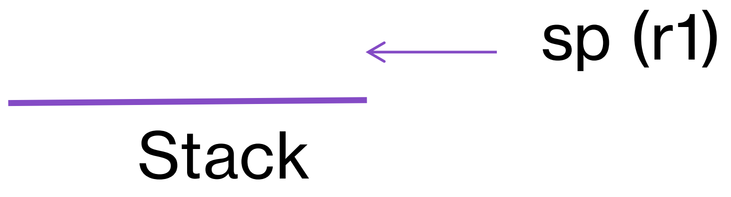
r2	
r3	2



Calling `rec_sum`

```
2         lcw r1 stack
6         loa r3 r0

8         lcw r2 rec_sum
12        cal r2 r2

14        sto r3 r0
16        hlt


rec_sum
18        psh r2
22        bne r3 r0 recurse
24        add r3 r0 0
26        brs done

…
```

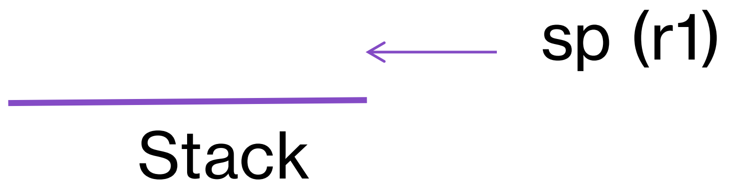
r2	18
r3	2



Calling `rec_sum`

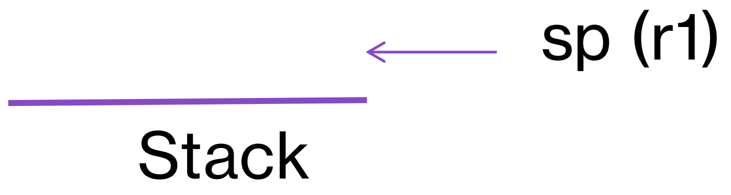
```
2         lcw r1 stack
6         loa r3 r0

8         lcw r2 rec_sum
12        cal r2 r2

14        sto r3 r0
16        hlt


rec_sum
18        psh r2
22        bne r3 r0 recurse
24        add r3 r0 0
26        brs done

...
```

r2	18
r3	2



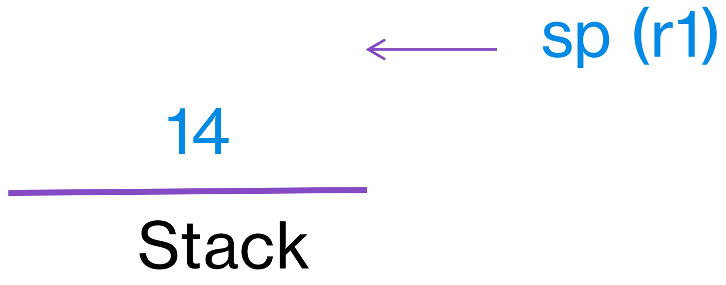
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	14
r3	2



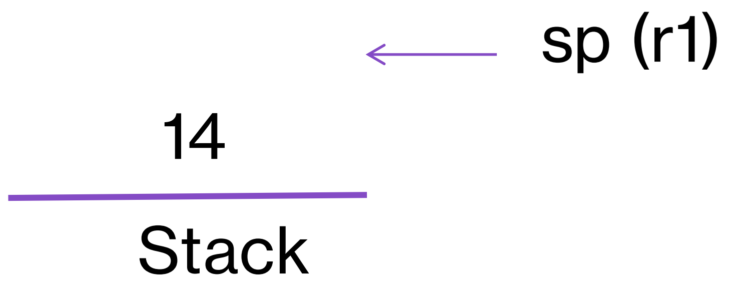
```
rec_sum
18      psh r2
22      bne r3 r0 recurse
24      add r3 r0 0
26      brs done

recurse
28      psh r3
32      sub r3 r3 1

34      lcw r2 rec_sum
38      cal r2 r2

40      pop r2
44      add r3 r3 r2
done
46      pop r2
50      jmp r2
```

r2	14
r3	2



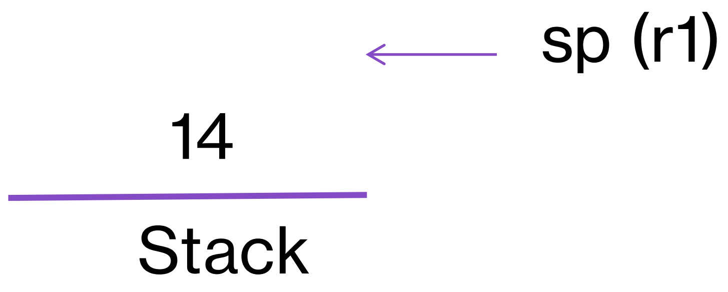
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	14
r3	2



```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

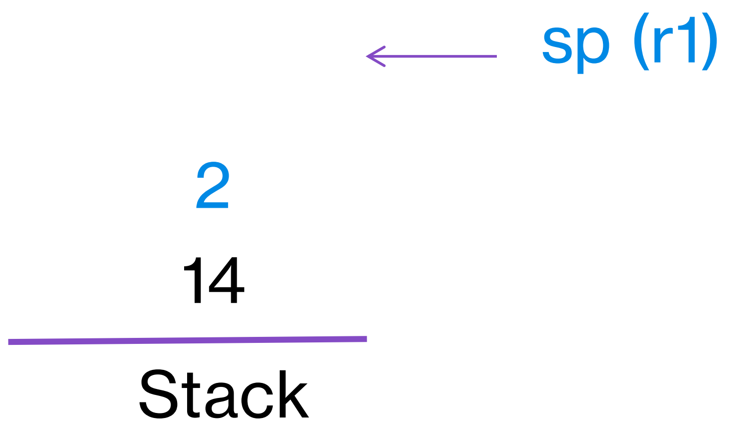
r2	14
r3	2



```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	14
r3	2

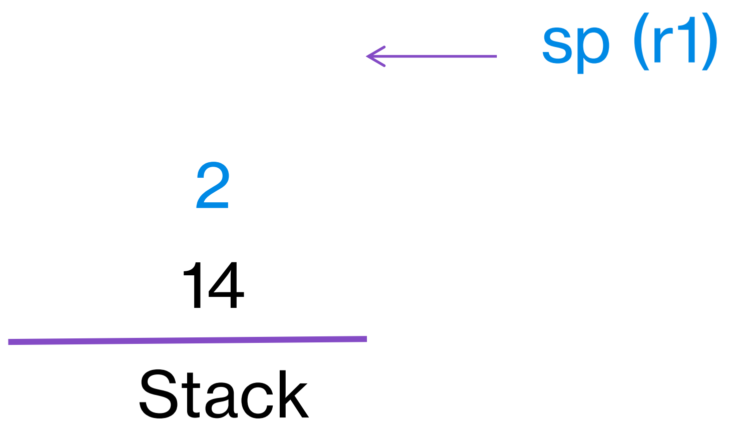
Why psh r3?



```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

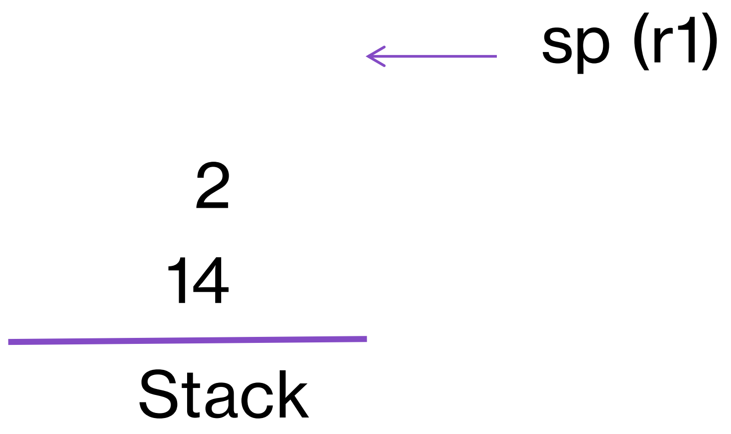
r2	14
r3	2

- We're about to make a function call
- The result of that call will go into r3 so that would erase its contents if not saved
- **n + rec_sum (n-1)**



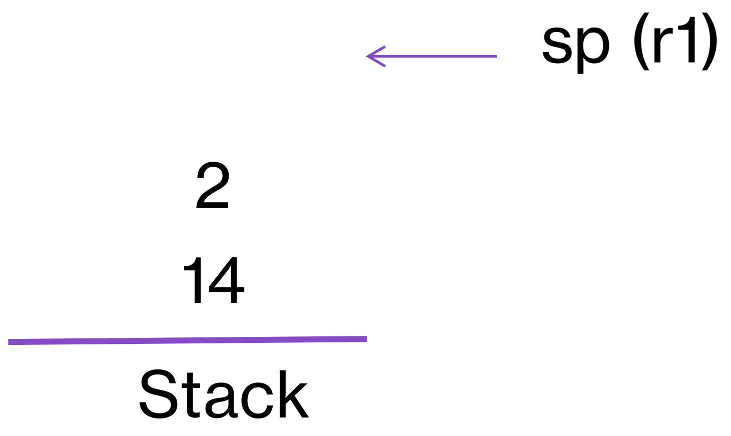
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	14
r3	2



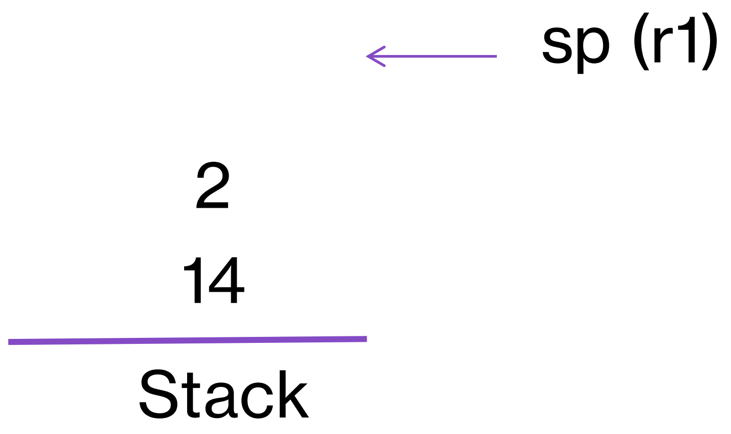
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	14
r3	1



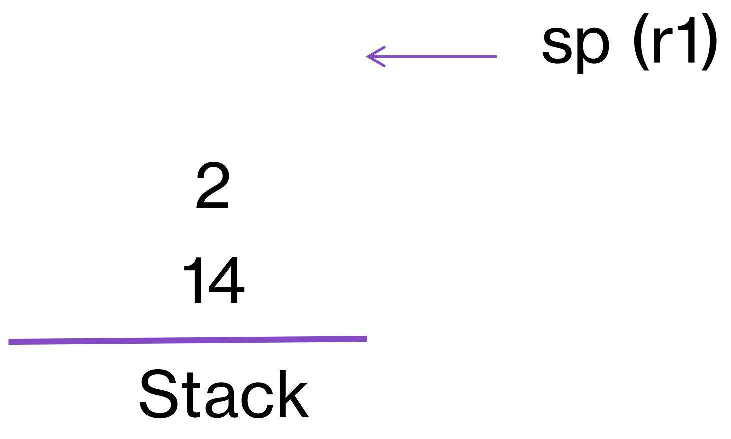
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	14
r3	1



```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	18
r3	1



```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	18
r3	1

← sp (r1)

2

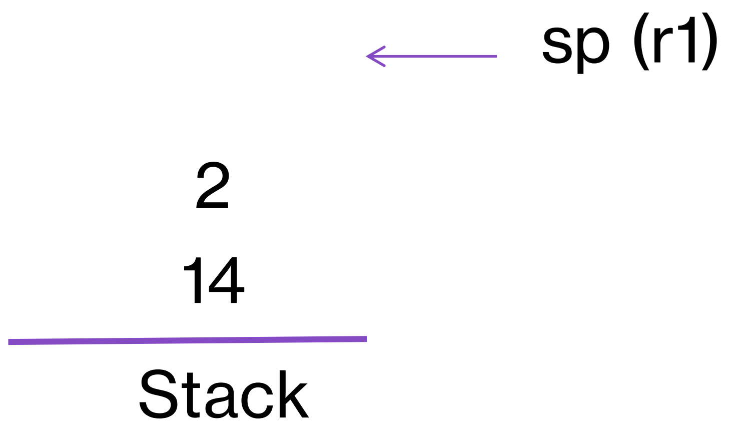
14

Stack

```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

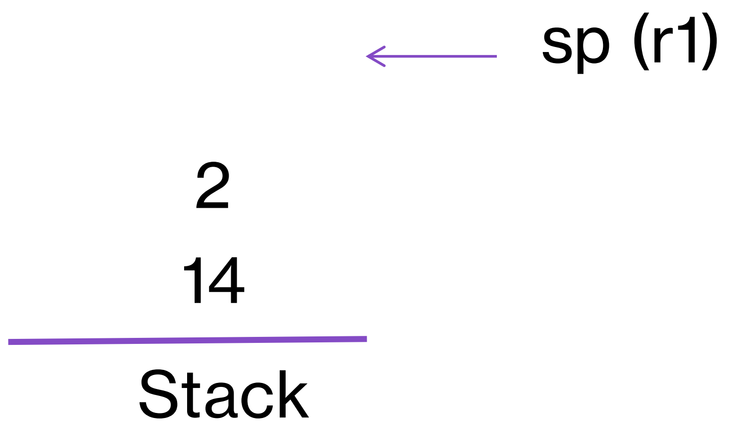
r2	18
r3	1

Make a recursive call:
rec_sum(1)



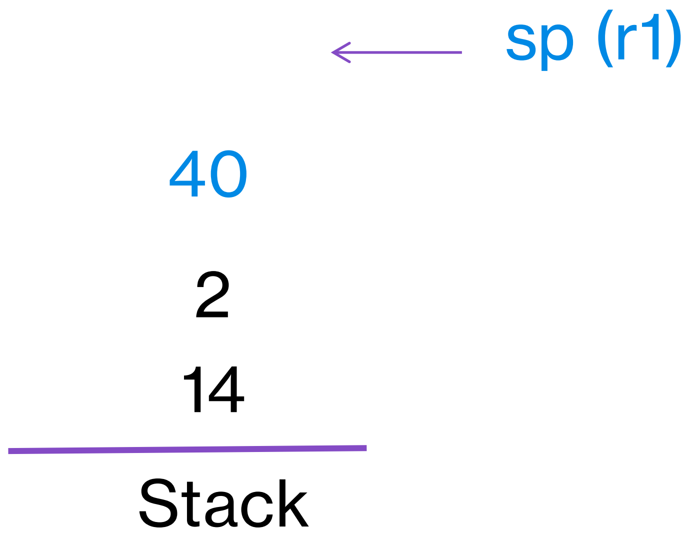
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	40
r3	1



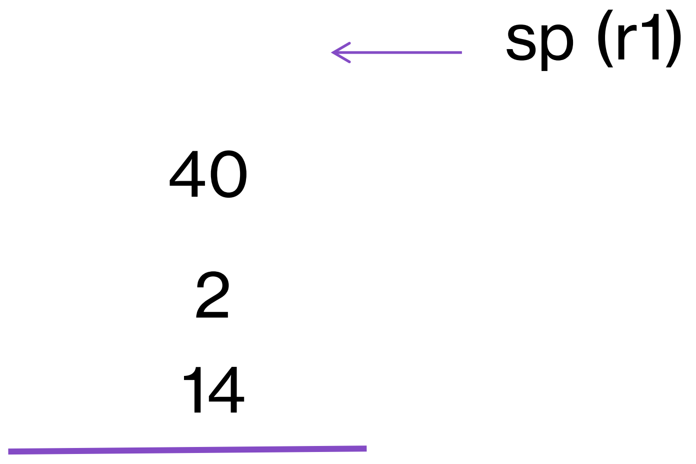
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	40
r3	1



```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

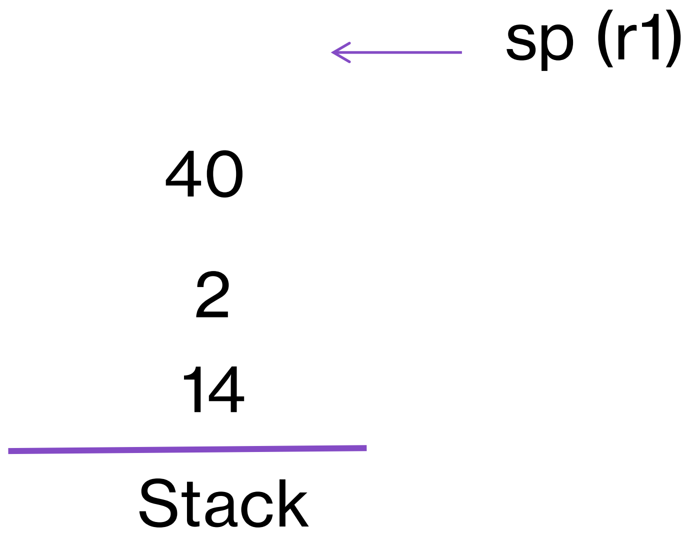
r2	40
r3	1



Stack

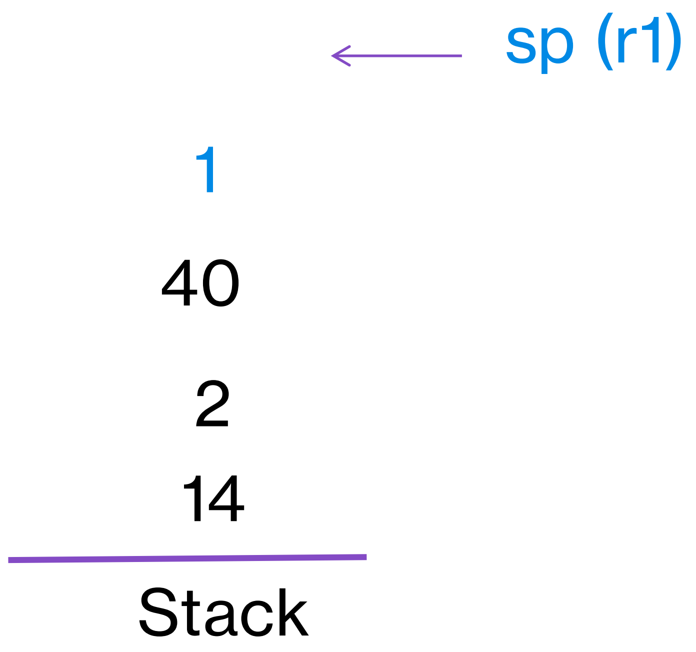
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	40
r3	1



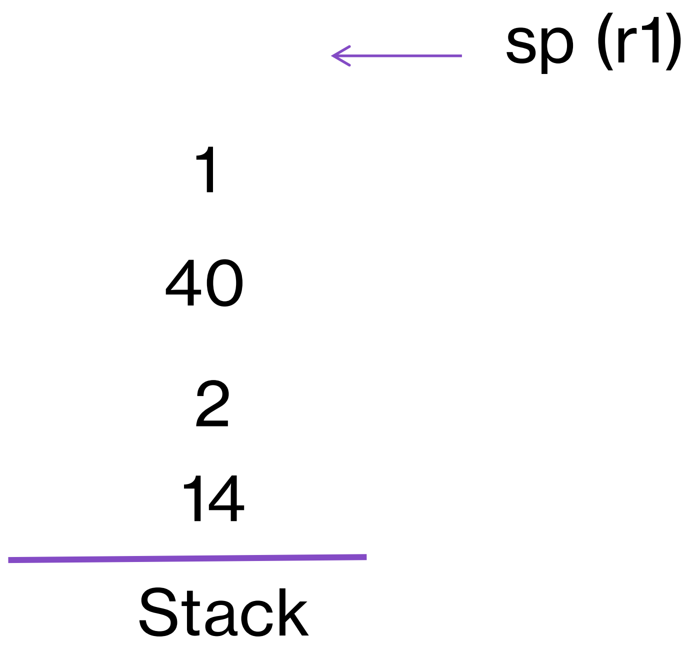
```
rec_sum
18        psh r2
22        bne r3 r0 recurse
24        add r3 r0 0
26        brs done

recurse
28        psh r3
32        sub r3 r3 1

34        lcw r2 rec_sum
38        cal r2 r2

40        pop r2
44        add r3 r3 r2
done
46        pop r2
50        jmp r2
```

r2	40
r3	1



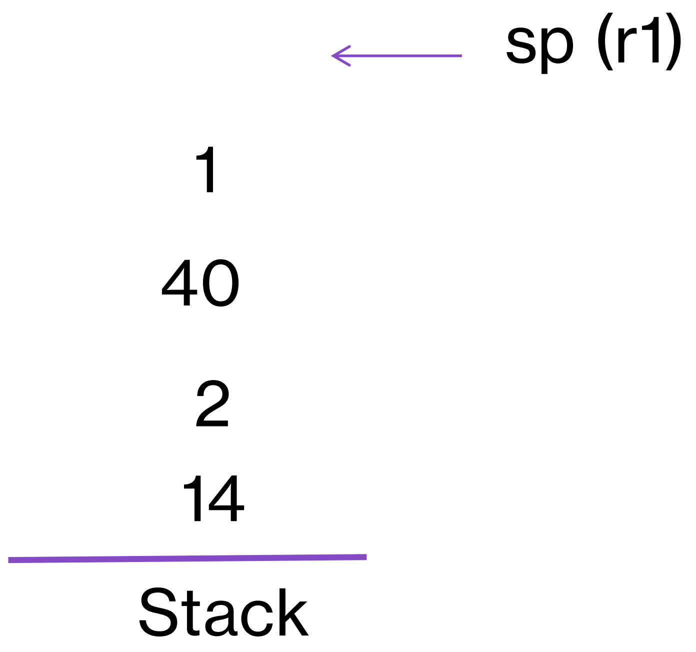
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	40
r3	1



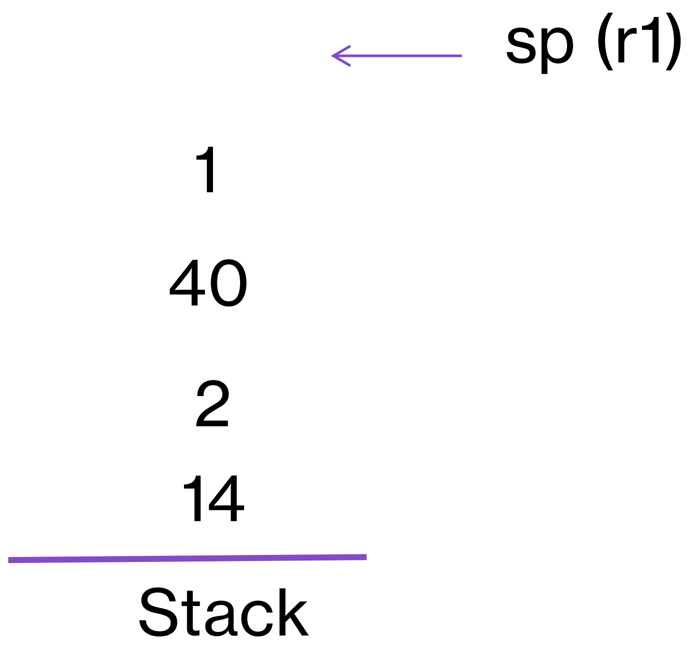
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	40
r3	0



```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

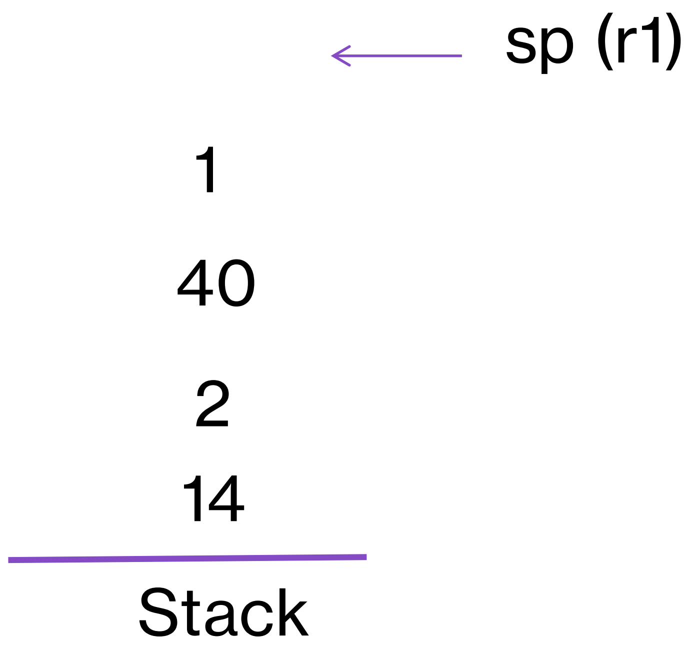
r2	18
r3	0



```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

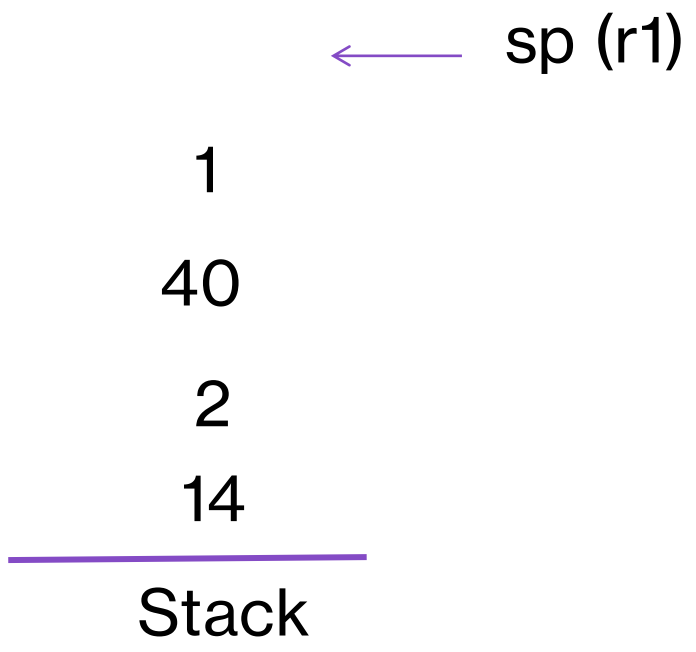
r2	18
r3	0

Make a recursive call:
rec_sum(0)



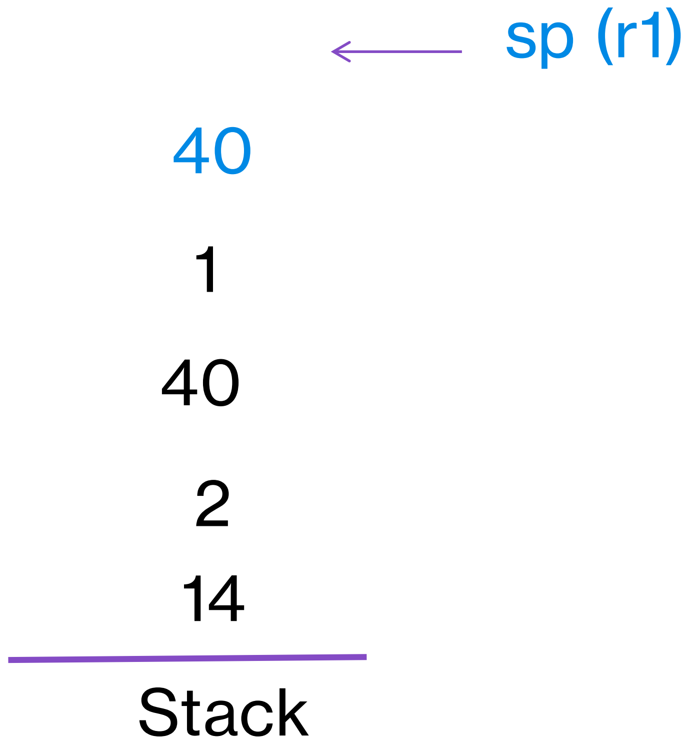
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	40
r3	0



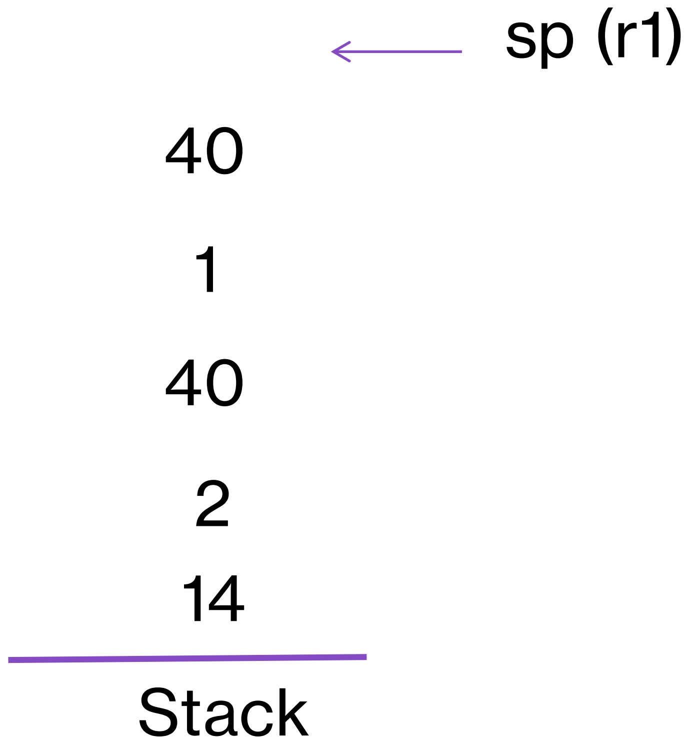
```
rec_sum
18         psh r2
22         bne r3 r0 recurse
24         add r3 r0 0
26         brs done

recurse
28         psh r3
32         sub r3 r3 1

34         lcw r2 rec_sum
38         cal r2 r2

40         pop r2
44         add r3 r3 r2
done
46         pop r2
50         jmp r2
```

r2	40
r3	0



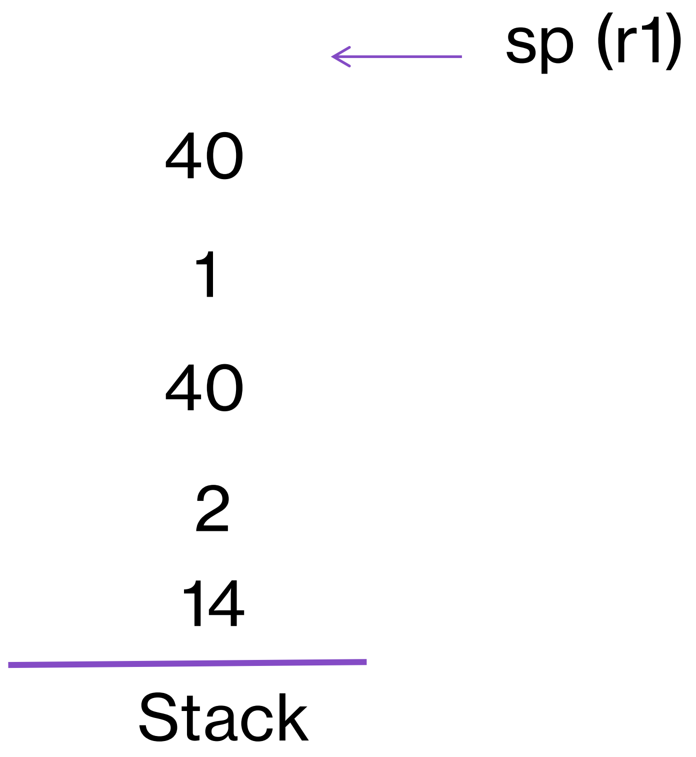
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	40
r3	0



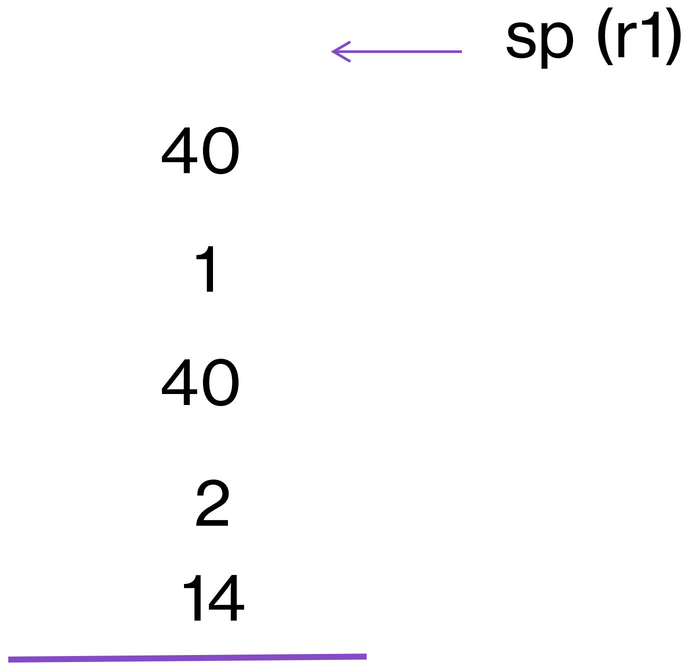
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	40
r3	0



```
rec_sum
18        psh r2
22        bne r3 r0 recurse
24        add r3 r0 0
26        brs done

recurse
28        psh r3
32        sub r3 r3 1

34        lcw r2 rec_sum
38        cal r2 r2

40        pop r2
44        add r3 r3 r2
done
46        pop r2
50        jmp r2
```

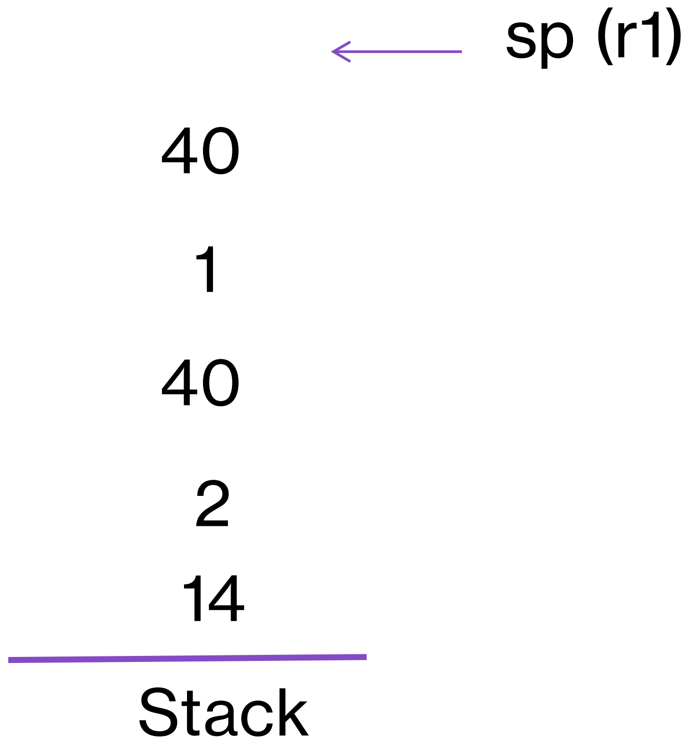
r2	40
r3	0



Stack

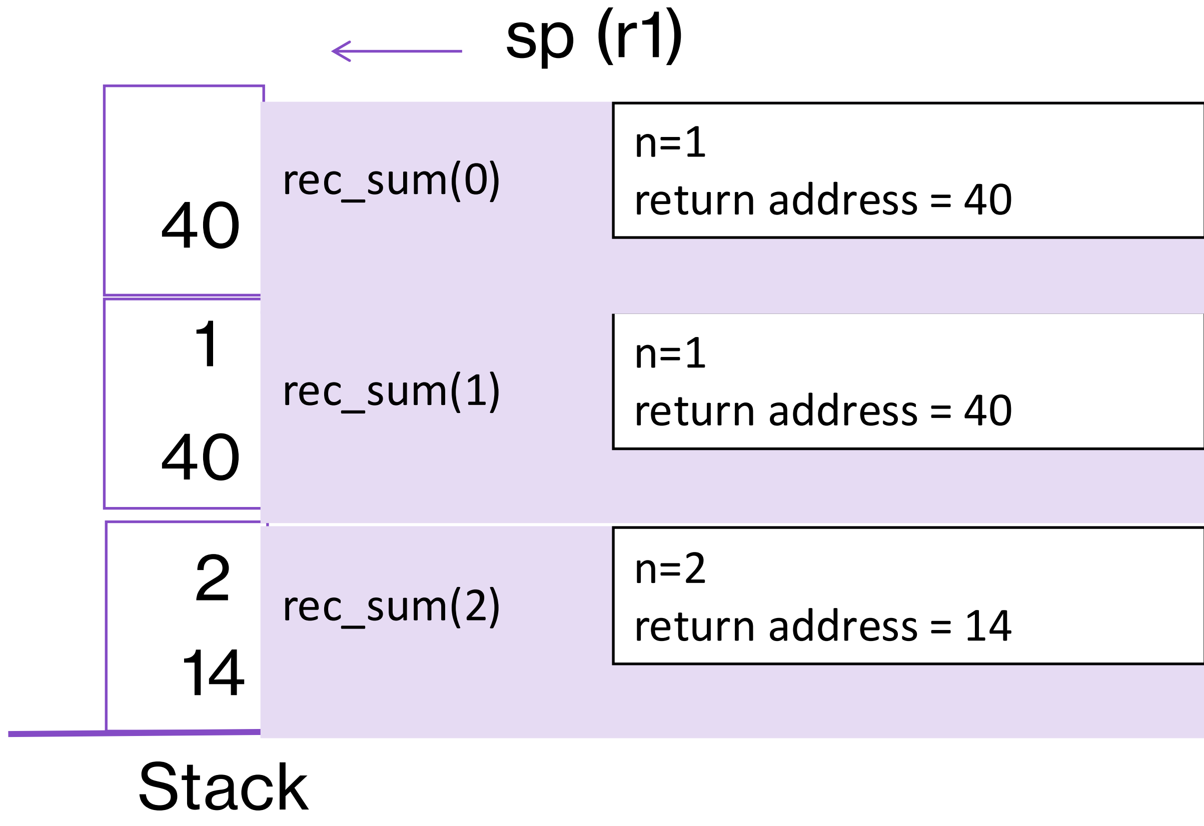
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	40
r3	0



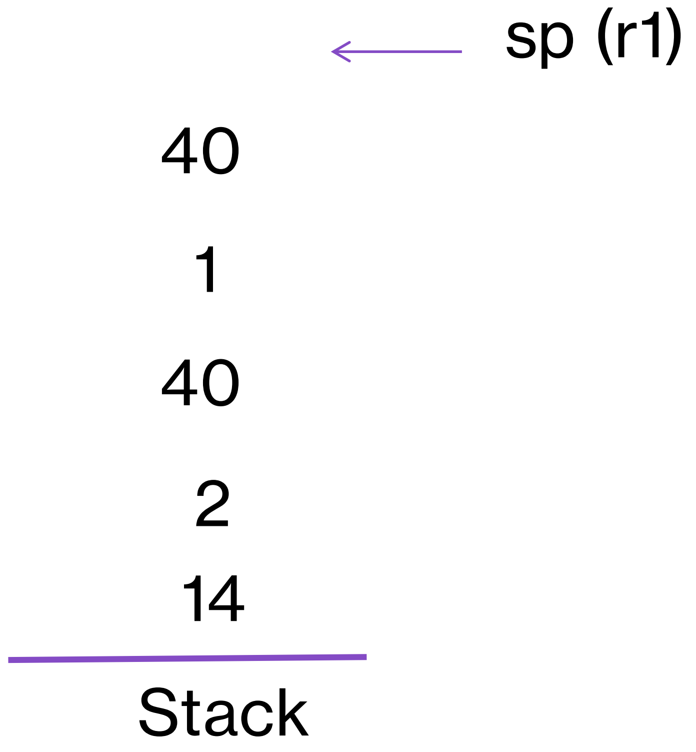
```
rec_sum
18        psh r2
22        bne r3 r0 recurse
24        add r3 r0 0
26        brs done

recurse
28        psh r3
32        sub r3 r3 1

34        lcw r2 rec_sum
38        cal r2 r2

40        pop r2
44        add r3 r3 r2
done
46        pop r2
50        jmp r2
```

r2	40
r3	0



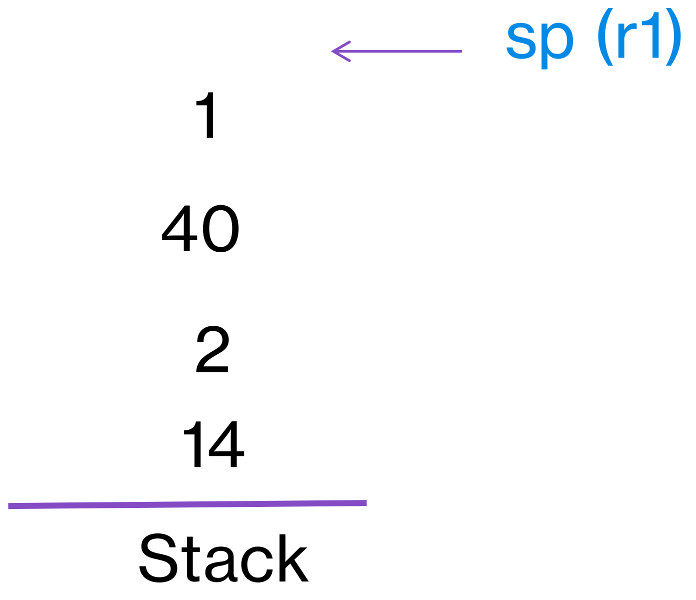
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	40
r3	0



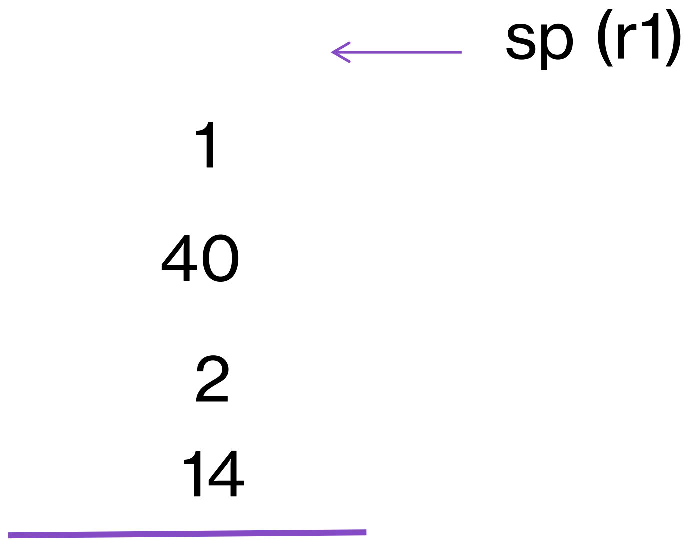
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	40
r3	0



```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	40
r3	0



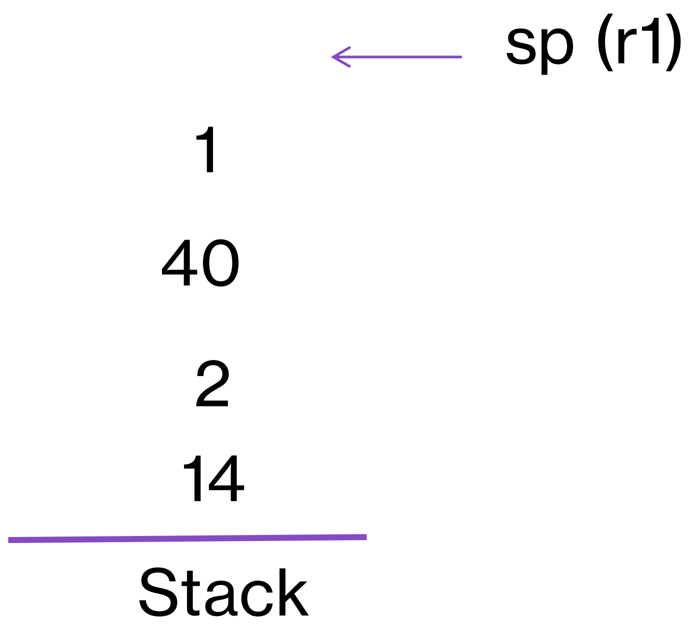
Stack

Returning with answer in r3

```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	40
r3	0

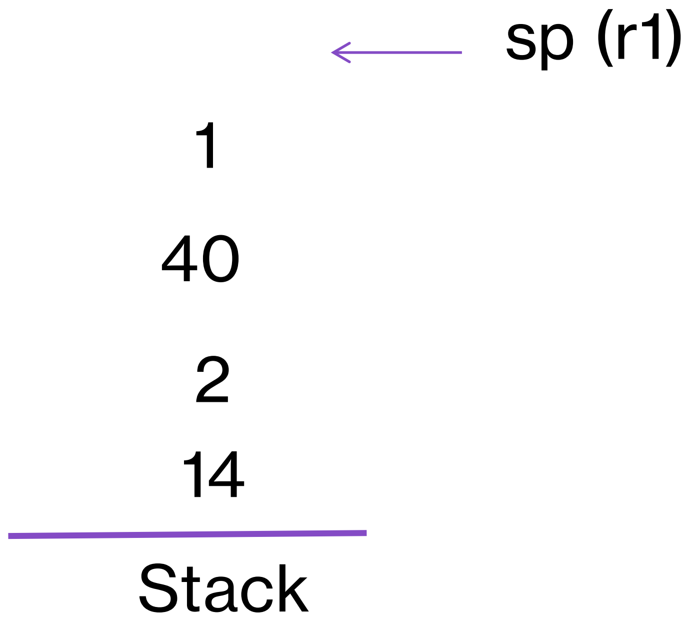
Why are we doing this?



```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

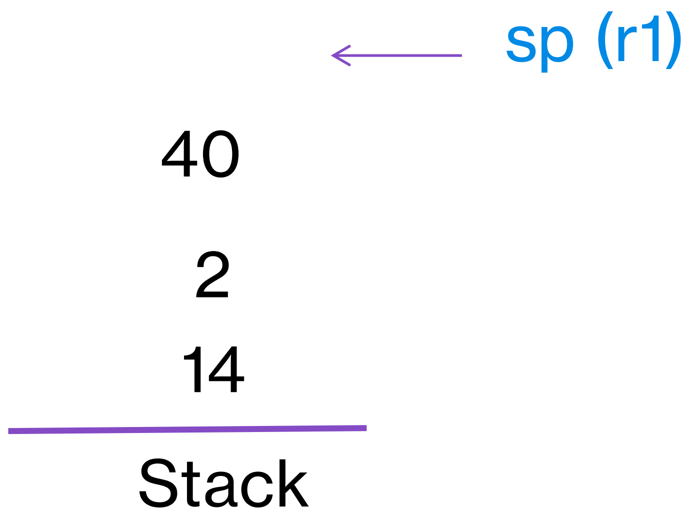
r2	40
r3	0

- Need to calculate n + rec_sum (n-1)
- Saved n on the stack



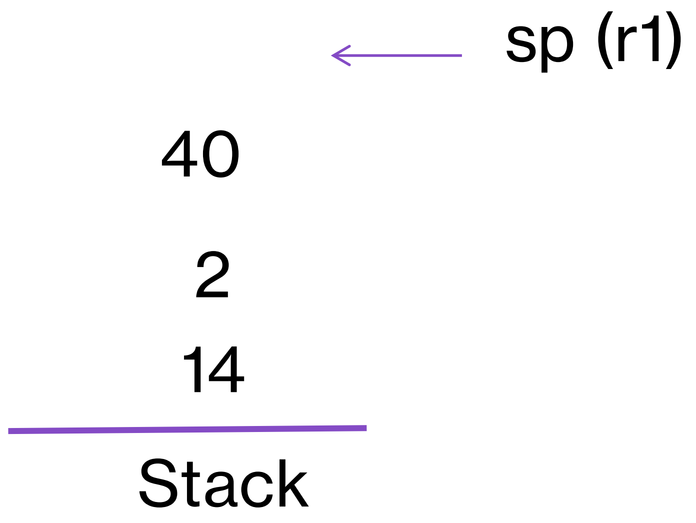
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	1
r3	0



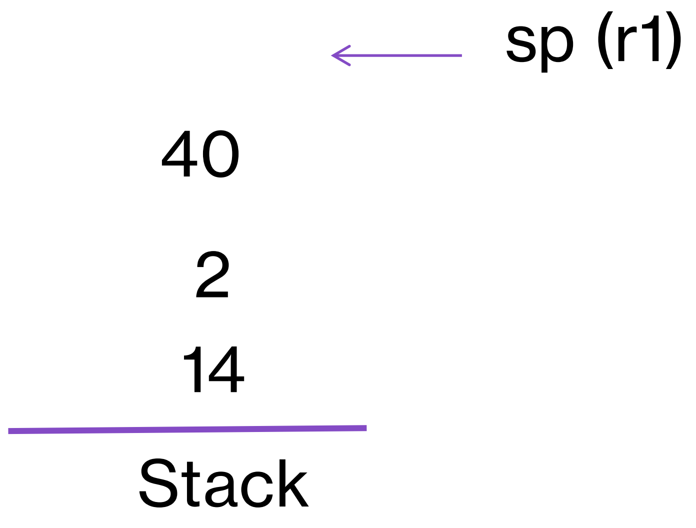
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	1
r3	0



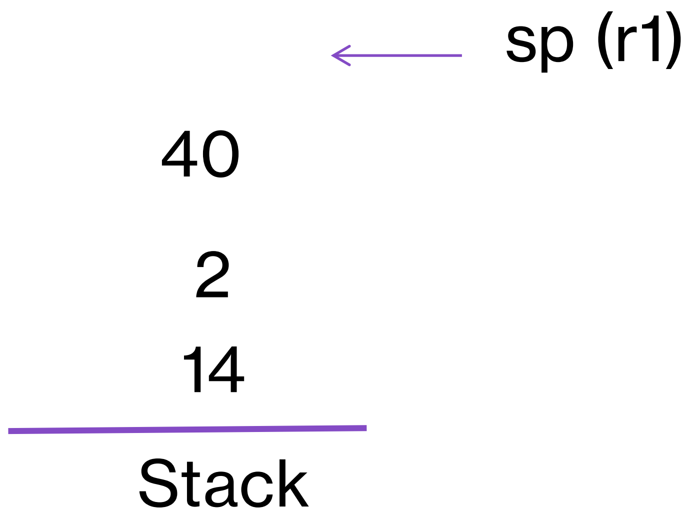
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	1
r3	1



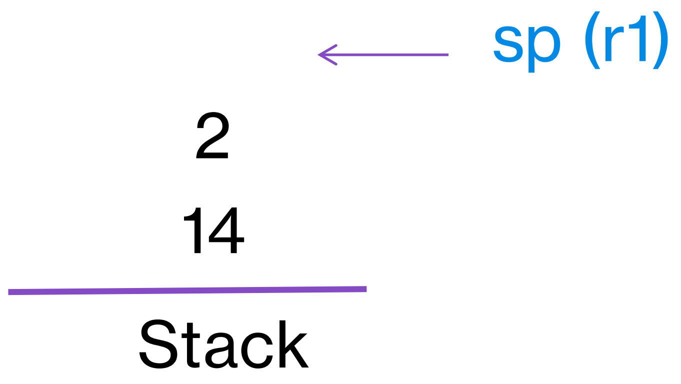
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	1
r3	1



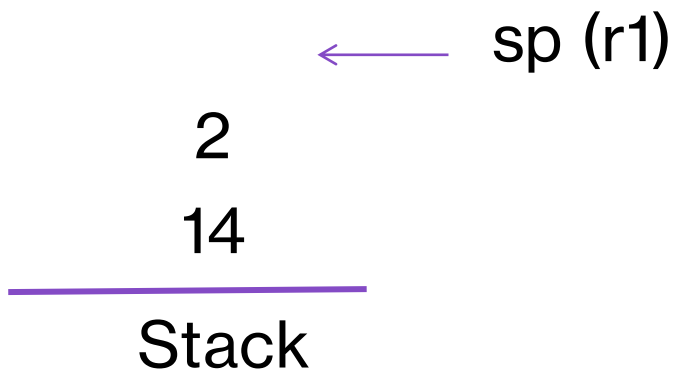
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	40
r3	1



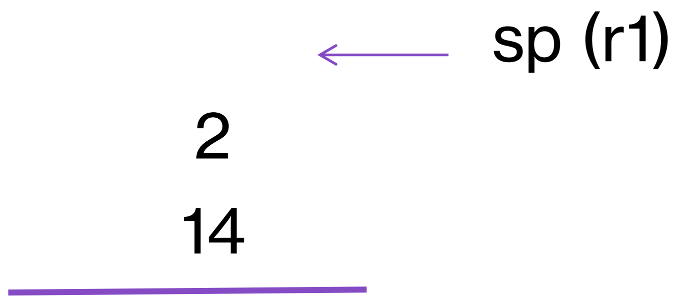
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	40
r3	1



```
rec_sum
18        psh r2
22        bne r3 r0 recurse
24        add r3 r0 0
26        brs done

recurse
28        psh r3
32        sub r3 r3 1

34        lcw r2 rec_sum
38        cal r2 r2

40        pop r2
44        add r3 r3 r2
done
46        pop r2
50        jmp r2
```

r2	40
r3	1

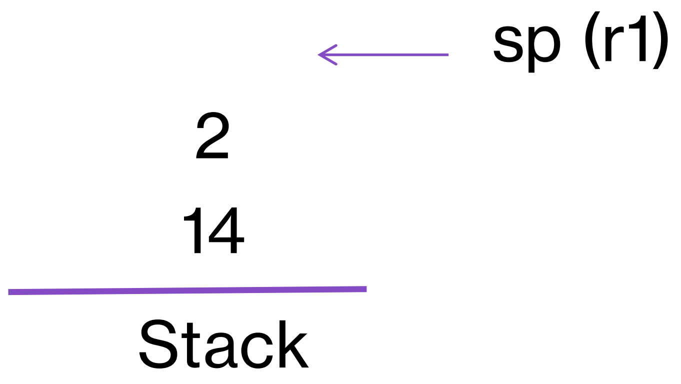


Stack

Returning with answer in r3

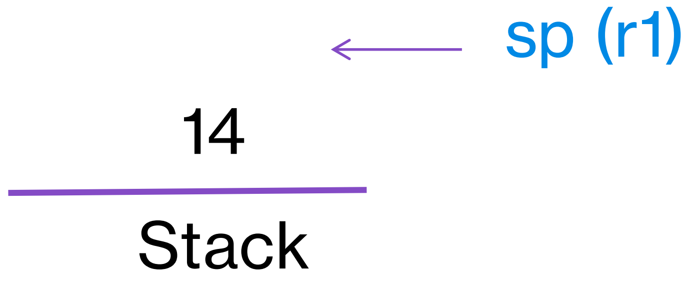
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	40
r3	1



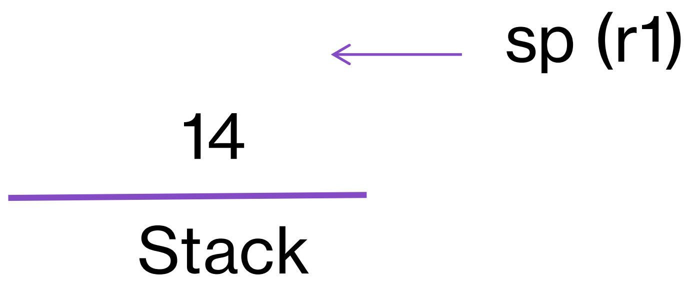
```
rec_sum
18        psh r2
22        bne r3 r0 recurse
24        add r3 r0 0
26        brs done

recurse
28        psh r3
32        sub r3 r3 1

34        lcw r2 rec_sum
38        cal r2 r2

40        pop r2
44        add r3 r3 r2
done
46        pop r2
50        jmp r2
```

r2	2
r3	1



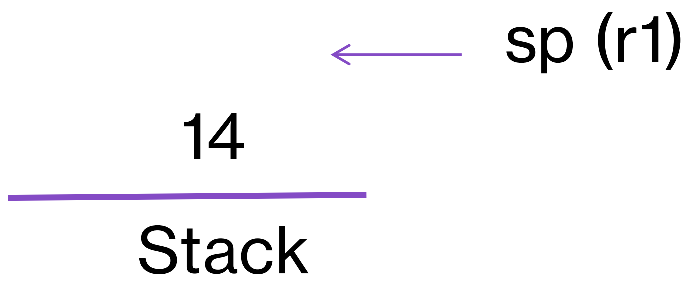
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	2
r3	1



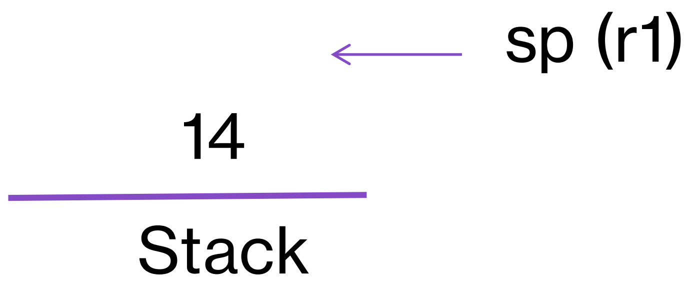
```
rec_sum
18        psh r2
22        bne r3 r0 recurse
24        add r3 r0 0
26        brs done

recurse
28        psh r3
32        sub r3 r3 1

34        lcw r2 rec_sum
38        cal r2 r2

40        pop r2
44        add r3 r3 r2
done
46        pop r2
50        jmp r2
```

r2	2
r3	3



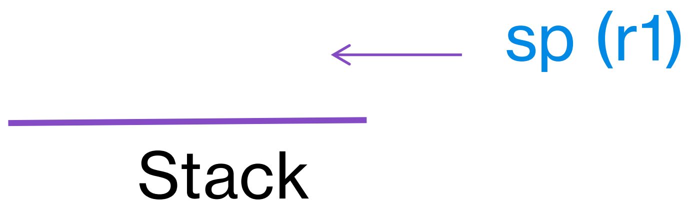
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	2
r3	3



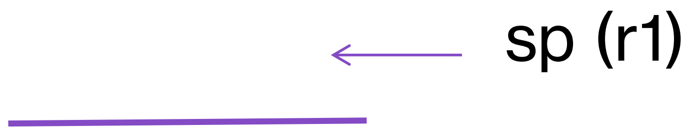
```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	14
r3	3



```
rec_sum
18          psh r2
22          bne r3 r0 recurse
24          add r3 r0 0
26          brs done

recurse
28          psh r3
32          sub r3 r3 1

34          lcw r2 rec_sum
38          cal r2 r2

40          pop r2
44          add r3 r3 r2
done
46          pop r2
50          jmp r2
```

r2	14
r3	3

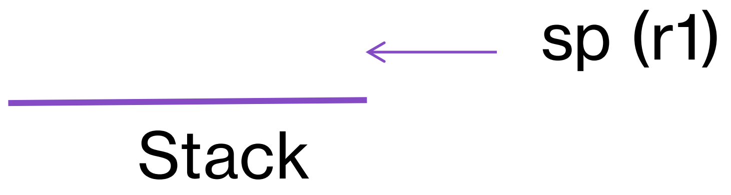


Stack

Returning with answer in r3

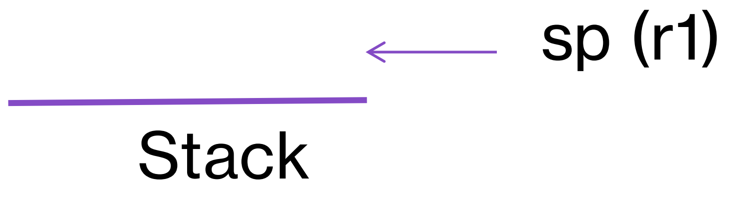
r2	14
r3	3

```
2         lcw r1 stack
6         loa r3 r0

8         lcw r2 rec_sum
12        cal r2 r2

14        sto r3 r0
16        hlt


rec_sum
18        psh r2
22        bne r3 r0 recurse
24        add r3 r0 0
26        brs done

…
```



```
2         lcw r1 stack
6         loa r3 r0

8         lcw r2 rec_sum
12        cal r2 r2

14        sto r3 r0
16        hlt


rec_sum
18        psh r2
22        bne r3 r0 recurse
24        add r3 r0 0
26        brs done

…
```

r2	14
r3	3



Python modules

- **Module**: a collection of functions and variables allowing us to share code with others.
- For example, there is a module called `math` that has many math functions you might want.
  - We can look at the documentation for this module by going to https://docs.python.org/3/ and searching `math` which will lead you to https://docs.python.org/3/library/math.html#module-math
- Some examples of useful functions and variables in the `math` module:
  - Variations of logs with different bases (e.g., `log`, `log2`, `log10`)
  - `sqrt` for calculating the square root
  - trigonometric functions (e.g., `cos`, `sin`, `tan`)
  - Constants (e.g., `pi`, `e`, etc.)

Importing modules

- If we want to use a module, we need to tell the program to "import" it in our program.
- Code to import all functions and variables from `math` module:
    - `from math import *`
- This statement has multiple components:
    - `from` is a keyword,
    - `math` is the name of the module,
    - `import` loads the module into our program,
    - `*` means everything, i.e. all functions and variables included in the `math` module.
- Just replace `math` with the name of the module you want to import.

`random` module

- The `random` module generates pseudo-random numbers.
- If you want truly random numbers, check out http://www.random.org/

Useful `random` functions

- `random()` - returns a random float between 0 and 1.
- `uniform(a, b)` - returns a random float between a and b.
- `randint(a, b)` - returns a random integer between a and b.

Importing only one function

- Let's say we only want to use the randint function.
- Rather than importing everything (*) we will be specific:
  - from random import randint

- What would the following piece of code do?

```
for i in range(100):
    print(randint(0,10))
```

- Prints 100 random integers between 0 and 10 (inclusive)

turtle module

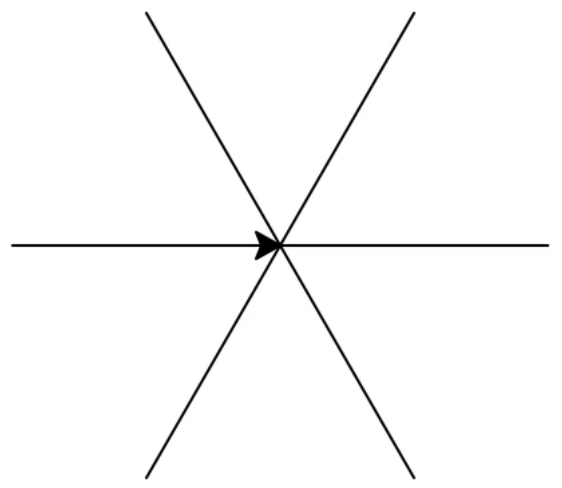
- The turtle module controls the movements of a "turtle" (really, an arrow), such as:
  - forward(distance): Move the turtle forward by the specified distance.
  - backward(distance): Move the turtle backward by distance, opposite to the direction the turtle is headed without changing the turtle’s heading.
  - right(angle): Turn the turtle right by angle units.
  - left(angle): Turn turtle left by angle units.
  - ...and many others. Check the documentation and more practice in the upcoming assignment.
- As the turtle moves, it draws a line, so by controlling it we can draw on the screen!

Open VS Code

- Go to the Python shell at the bottom
- python3
- >>> from turtle import *

A star with a different number of spokes

```
def asterisk_star(length, spokes):
    angle = 360 / spokes
    for i in range(spokes):
        forward(length)
        backward(length)
        right(angle)
```

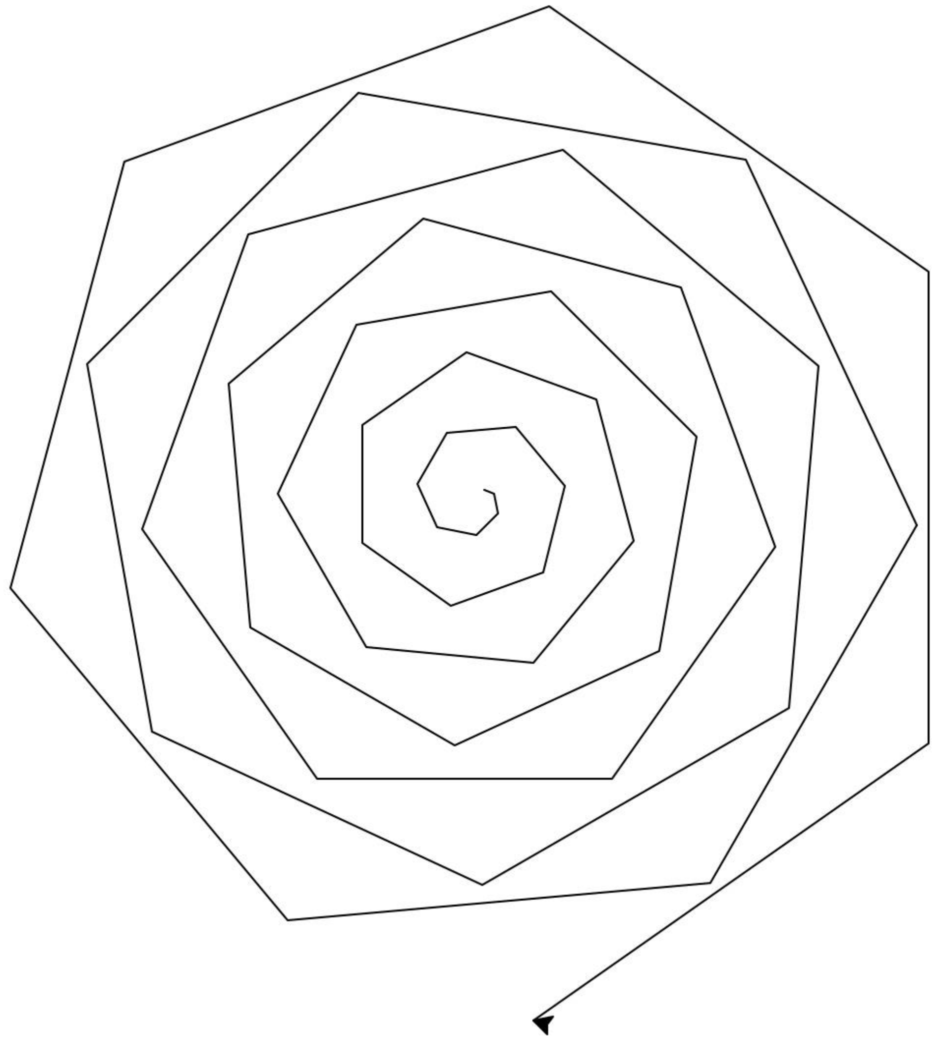


mystery1 function

```
def mystery1():
    for i in range(50):
        forward(i * 5)
        right(55)
```

simple_spiral function

```
def simple_spiral():
    for i in range(50):
        forward(i * 5)
        right(55)
```

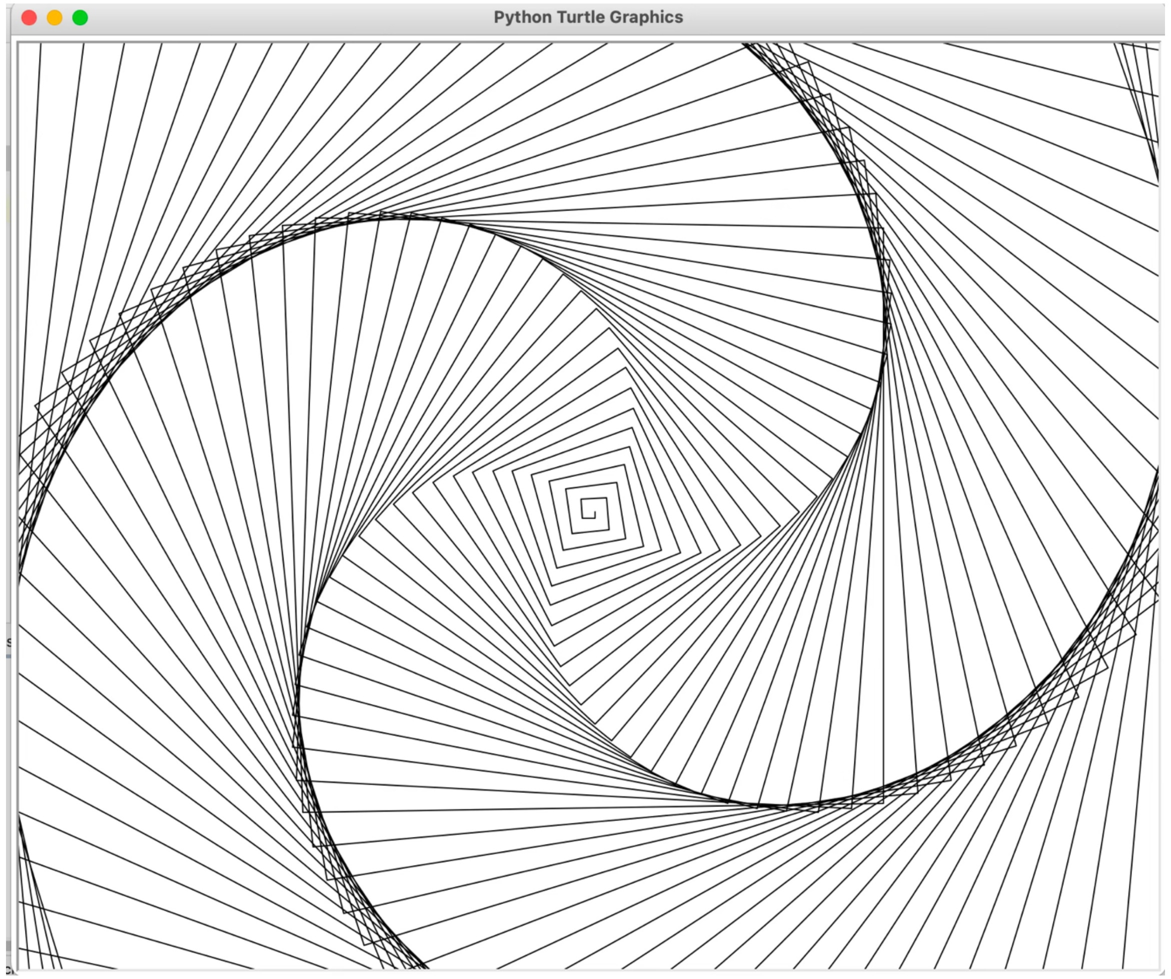


mystery2 function

```
def mystery2(par1, par2):
    for i in range(par1):
        forward(i * 5)
        right(par2)
```

spiral function

```
def spiral(sides, angle):
   for i in range(sides):
      forward(i * 5)
      right(angle)
```

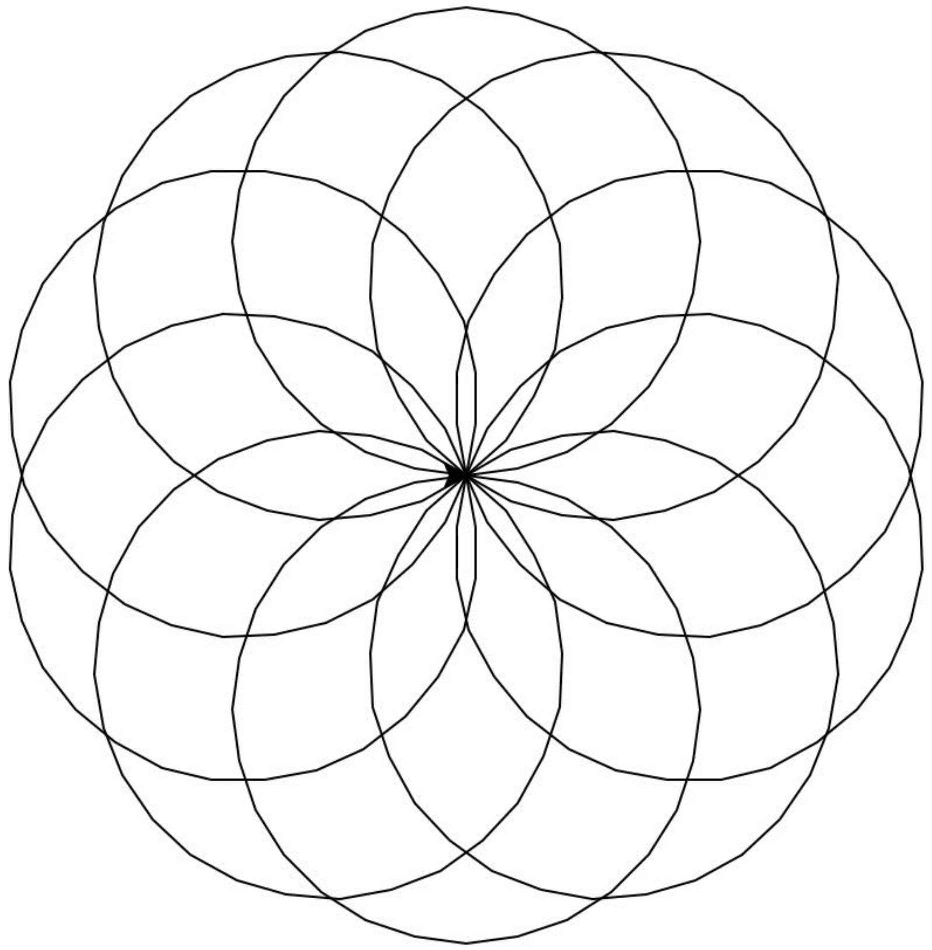


mystery3 function

```
def mystery3(par1, par2):
    var = 360 / par2


    for i in range(par2):
        circle(par1)
        right(var)
```

rotating_circles function

```
def rotating_circles(radius, num):
    angle = 360 / num


    for i in range(num):
        circle(radius)
        right(angle)
```

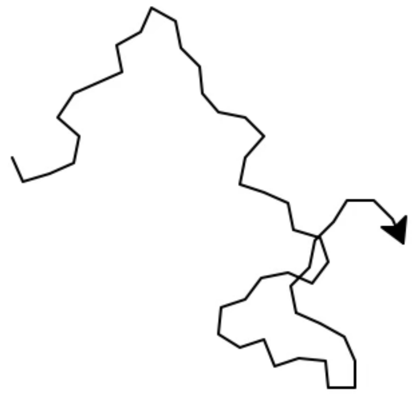


mystery4 function

```
def mystery4(par1, par2):
    for i in range(par1):
        var = randint(-90, 90)
        right(var)
        forward(par2)
```

walk function

```
def walk(num_steps, step_size):
    for i in range(num_steps):
        angle = randint(-90, 90)
        right(angle)
        forward(step_size)
```

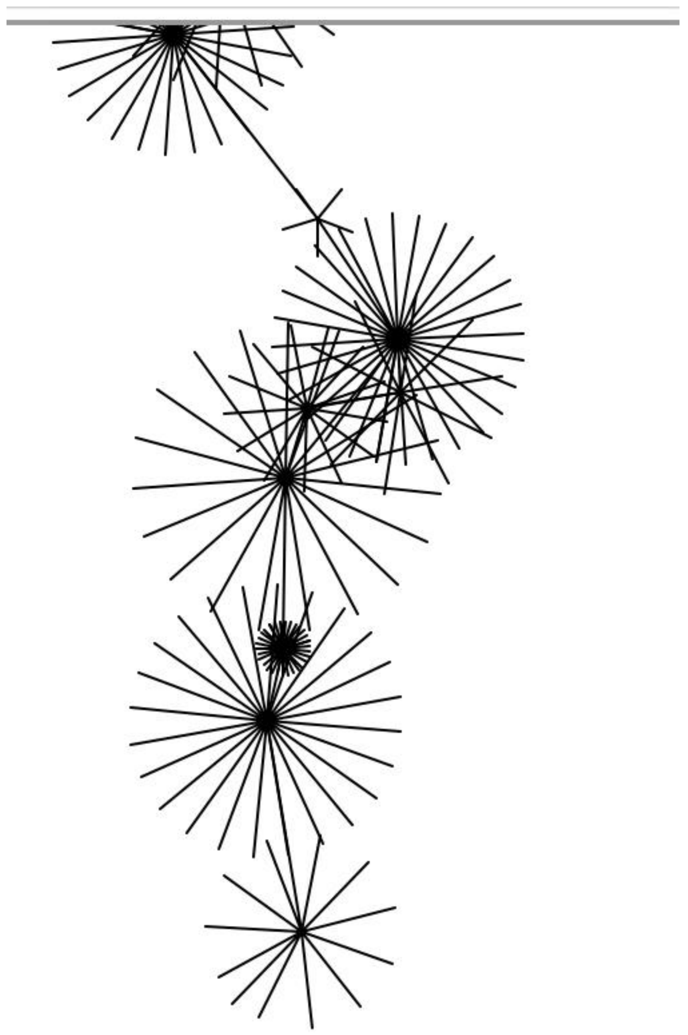


mystery5 function

```
def mystery5():
    for i in range(10):
        var1 = randint(5, 30)
        var2 = randint(10, 60)
        var3 = randint(-90, 90)
        var4 = randint(20, 100)
        right(var3)
        forward(var4)
        asterisk_star(var2, var1)
```

pretty_picture function

```
def pretty_picture():
    for i in range(10):
        spokes = randint(5, 30)
        length = randint(10, 60)
        angle = randint(-90, 90)
        move = randint(20, 100)
        right(angle)
        forward(move)
        asterisk_star(length, spokes)
```

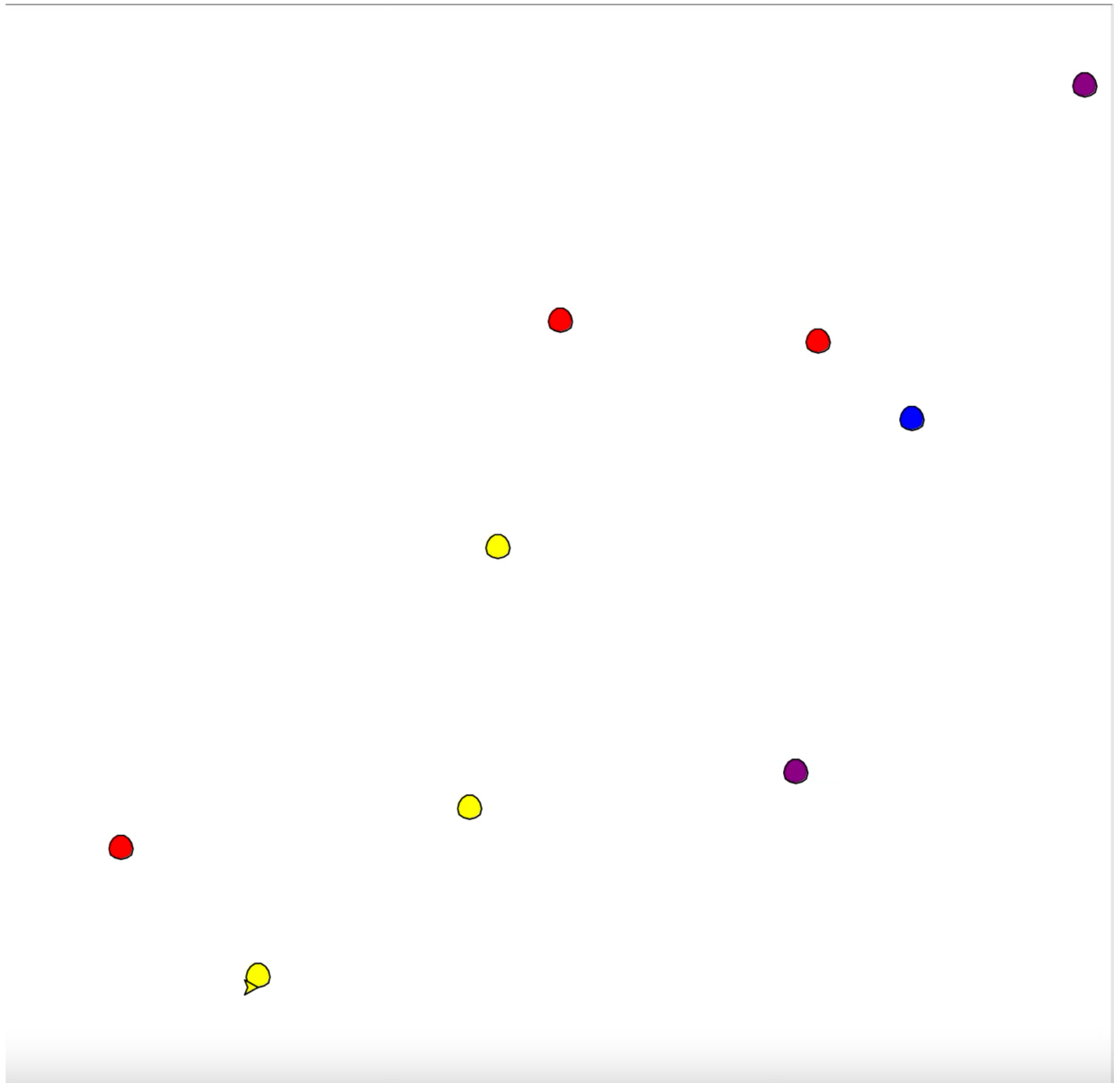


setcolor_random function

```
def setcolor_random():
    color = randint(1, 4)
    if color == 1:
        fillcolor("blue")
    elif color == 2:
        fillcolor("purple")
    elif color == 3:
        fillcolor("red")
    else:  # color == 4
        fillcolor("yellow")
```

add_circles function

```
def add_circles(number):
    x_range = int(window_width() / 2)
    y_range = int(window_height() / 2)
    for i in range(number):
        x = randint(-x_range, x_range)
        y = randint(-y_range, y_range)
        setcolor_random()
        pu()
        goto(x, y)
        pd()
        begin_fill()
        circle(8)
        end_fill()
```



Revisiting creating a spiral

```
def spiral(length, levels):
    if levels == 0:
        dot()
    else:
        forward(length)
        left(30)
        spiral(0.95 * length, levels – 1)
```

Revisiting spiral function

- Let’s assume I called spiral(80, 50)
- When does it stop?
  - When levels = 0.
    - We put a dot there to make it explicit.
- Repeat 50 times:
  - forward length
  - left 30
  - reduce length by 5%

```
def spiral(length, levels):
   if levels == 0:
      dot()
   else:
      forward(length)
      left(30)
      spiral(0.95 * length, levels – 1)
```

Revisiting spiral function

- What if we wanted to end up back at the starting point, but not pick the pen up?
- We could trace our steps backwards.

```
def spiral(length, levels):
    if levels == 0:
        dot()
    else:
        forward(length)
        left(30)
        spiral(0.95 * length, levels – 1)
        right(30)
        backward(length)
```

broccoli_demo function

1. Define what the header function is:
   - broccoli(x, y, length, angle)
2. Define the recursive case:
   - broccoli is a line with three other broccolis at the end:
     - one directly straight out
     - one 20 degrees to the left
     - one 20 degrees to the right
   - the three other broccolis should be smaller/shorter than the current

broccoli_demo function

3. Define the base case:
   - in each case, the length of the broccoli to be drawn gets shorter.
   - We stop at `length < 10` and place a yellow dot
4. put it all together!
   - Check the base case first:
     - if `length < 10`
       - Draw a yellow dot.
   - Otherwise:
     - draw three smaller broccolis at different angles.

- `new_x` and `new_y` are the ending coordinates of the line being drawn. We save them because after the first recursive call to broccoli the turtle won't be in the same place.

Code

rec_sum.a51

turtle-examples.py

turtle-recursion.py