FND14

iterative.c source code

```c
#include <stdio.h>
int fac2(int n)
{
    printf("Top of the stack: %p\n", &n);
    int val = 1;
    // printf("&val: %p\n", &val);
    if (n == 0)
    {
        val = 1;
        return val;
    }
    while (n > 0)
    {
        val *= n;
        n--;

    }

    return val;
}
int main(int argc, char *argv[])
{
    int x = 5;
    int f = fac2(x);
    printf("Bottom of the stack: %p\n", &x);
    printf("%d\n", f); // RL1
}

```

iterative.c output when x = 5

```
PS C:\Users\momou\Desktop\University\CS 2263\ForNextDay\ForNextDay()14> gcc -o iterative iterative.c
PS C:\Users\momou\Desktop\University\CS 2263\ForNextDay\ForNextDay()14> .\iterative
Top of the stack: 0061FF00
Bottom of the stack: 0061FF18
120
```

Memory consumption: 0061ff18 – 0061ff00 = 18

iterative.c output when x = 10

```
PS C:\Users\momou\Desktop\University\CS 2263\ForNextDay\ForNextDay()14> gcc -o iterative iterative.c
PS C:\Users\momou\Desktop\University\CS 2263\ForNextDay\ForNextDay()14> .\iterative
Top of the stack: 0061FF00
Bottom of the stack: 0061FF18
3628800
```

Memory consumption: 0061ff18 – 0061ff00 = 18

iterative.c output when x = 15

```
PS C:\Users\momou\Desktop\University\CS 2263\ForNextDay\ForNextDay()14> gcc -o iterative iterative.c
PS C:\Users\momou\Desktop\University\CS 2263\ForNextDay\ForNextDay()14> .\iterative
Top of the stack: 0061FF00
Bottom of the stack: 0061FF18
2004310016
```

Memory consumption: 0061ff18 – 0061ff00 = 18

recursive.c source code

```c
#include <stdio.h>
int fac(int n)
{
    printf("Top of the stack: %p\n", &n);
    int val;
    if (n == 0)
    {
        val = 1;
        return val;
    }
    val = n * fac(n - 1);
    return val; // RL2
}
int main(int argc, char *argv[])
{
    int x = 5;
    int f = fac(x);
    printf("Bottom of the stack: %p\n", &x);
    printf("%d\n", f); // RL1
}
```

recursive.c output when x = 5

```
PS C:\Users\momou\Desktop\University\CS 2263\ForNextDay\ForNextDay()14> gcc -o recursive recursive.c
PS C:\Users\momou\Desktop\University\CS 2263\ForNextDay\ForNextDay()14> ./recursive
Top of the stack: 0061FE10
Bottom of the stack: 0061FF18
120
```

Memory consumption: 0061ff18 – 0061fe10 =108

recursive.c output when x = 10

```
PS C:\Users\momou\Desktop\University\CS 2263\ForNextDay\ForNextDay()14> gcc -o recursive recursive.c
PS C:\Users\momou\Desktop\University\CS 2263\ForNextDay\ForNextDay()14> ./recursive
Top of the stack: 0061FD20
Bottom of the stack: 0061FF18
3628800
```

Memory consumption: 0061ff18 – 0061fd20= 1f8

recursive.c output when x = 15

```
PS C:\Users\momou\Desktop\University\CS 2263\ForNextDay\ForNextDay()14> gcc -o recursive recursive.c
PS C:\Users\momou\Desktop\University\CS 2263\ForNextDay\ForNextDay()14> ./recursive
Top of the stack: 0061FC30
Bottom of the stack: 0061FF18
2004310016
```

Memory consumption: 0061ff18 – 0061fc30 = 2E8

Using the iterative method was more efficient than using the recursive method. The memory consumption, on one hand, during iteration was the same regardless of the input because we did not add stack frames. On the other hand, input (value of x) did matter significantly. The more the value of x was, the more stack frames we had to add, leading to memory consumption.