In this assignment, you will experiment with pointers, recursions and an interesting mathematical conjecture, namely Collatz.

Collatz conjecture has not been proven yet even though it is seemingly a simple problem. Consider the following function:

$$f(x) = \begin{cases} 3x+1 & if\ x\ is\ odd \\ x/2 & if\ x\ is\ even \end{cases}$$

Starting with an integer x_0 the sequence $x_1 = f(x_0), x_2 = f(x_1), \ldots$ tends end up with a loop 4, 2, 1, 4, 2, 1… Conjecture says that any starting integer should send this sequence to the same loop. (See https://www.youtube.com/watch?v=094y1Z2wpJg for an overview of this conjecture).

Towards the analysis of this conjecture, you are asked to write a few C functions:

1. **(25 pts)** Write a recursive function to generate and return the sequence.

   ```
   void generate_sequence (int xs, int currentlen, int seqlen, int *seq)
   ```

   Where `xs` is the first element of the sequence, `seqlen` is the length of the sequence, `currentlen` is the length of the sequence in the time of the function called, `*seq` is the array of the sequence.

2. **(30 pts)** Write a recursive function such that given any function and a starting integer check if the sequence ends up in a loop.

   ```
   void check_loop_iterative(void (*f)(?), int xs, int seqlen, int *loop,
   int *looplen)
   ```

   The first input is the function for which the sequence will be generated (fill its parameters declared with "?" above), starting from the first integer `xs` with a length of `seqlen`. The last two arguments will return the loop and its length. If no loop is found, the function will not update `*loop` array and return 0 with `looplen` variable.

   Note that the loop's length can be n/2 at most and 2 at least, you should search for a loop with every possible length (until you find one) in descending order.

3. **(20 pts)** To be used in the previous function, you are asked to write a function to check if a given sequence of numbers has a loop in it.

```
int has_loop(int *arr, int n, int looplen, int *ls, int *le)
```

The function will receive a sequence starting from `arr[0]` and ending at `arr[n-1]`. It will check if there are any loops with a length of `looplen`. The first occurrence of the loop will be output with parameters ls and le.

This is not a void function, it will return 1 if a loop exists, 0 otherwise.

For example,

{15, 7, 4, 2, 1, 4, 2, 1}

has a loop starting at `arr[2]` and ending at `ar[4]`. In this case `ls=2` and `le=4`.

4. **(25 pts)** Write another recursive function that calculates the histogram of the first digits of the sequence generated by a given function.

```
void hist_of_firstdigits(void (*f)(?), int xs, int seqlen, int *h, int
digit)
```

As before the first three parameters are for the generation of the sequence. `h` is the histogram of the first digits of the numbers that appear in the sequence. The size of this array will be 9 (you will ignore digit 0).

For the following sequence:

{340, 170, 85, 256, 128, 64, 32, 16, 8, 4, 2, 1, 4, 2, 1}

The first digits of each integer:

{3, 1, 8, 2, 1, 6, 3, 1, 8, 4, 2, 1, 4, 2, 1}

The resulting histogram (`*h`) would be:

{5,3,2,2,0,1,0,2,0}

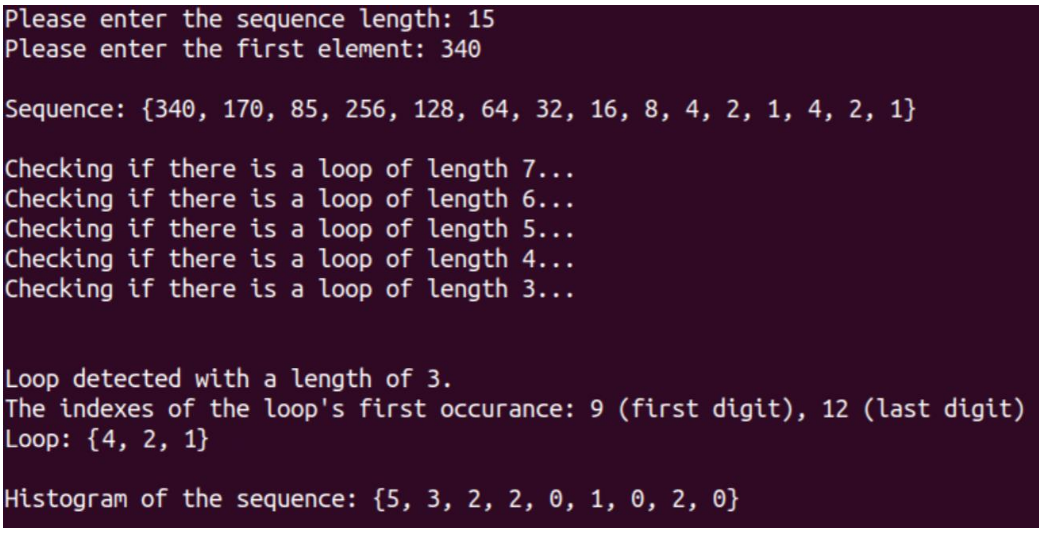
meaning that there are 0 numbers starting with digit 5 while there is only one number starting with digit 6.

Every time this function is being called; it should find the total numbers of integers starting with `digit`.

Program flow:

- In main, take the first element and the size of the sequence as inputs from the user (assume that the inputs will be proper, i.e., positive integers).
- In main, call the check_loop_iterative function.
- In check_loop_iterative function, generate & print the sequence (print it only once). Also, check if the sequence has a loop. If there is a loop, print its size and the indexes of its first occurrence. Do not print the loop itself in this function.
- In main, print the loop (if any). If there is no loop, print this information.
- In main, call the hist_of_firstdigits function.
- Print the histogram array in main.

Expected outputs (first two rows are inputs, the rest is output only)



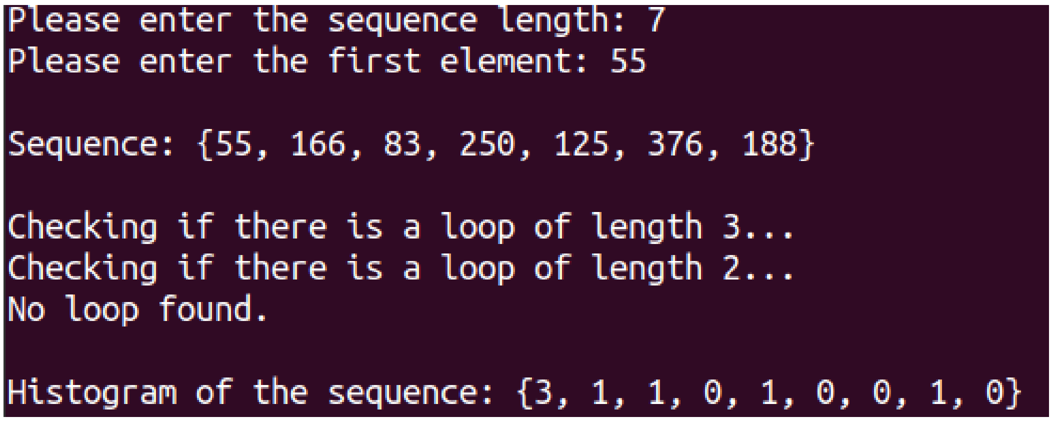
```
Please enter the sequence length: 15
Please enter the first element: 340

Sequence: {340, 170, 85, 256, 128, 64, 32, 16, 8, 4, 2, 1, 4, 2, 1}

Checking if there is a loop of length 7...
Checking if there is a loop of length 6...
Checking if there is a loop of length 5...
Checking if there is a loop of length 4...
Checking if there is a loop of length 3...


Loop detected with a length of 3.
The indexes of the loop's first occurance: 9 (first digit), 12 (last digit)
Loop: {4, 2, 1}

Histogram of the sequence: {5, 3, 2, 2, 0, 1, 0, 2, 0}
```

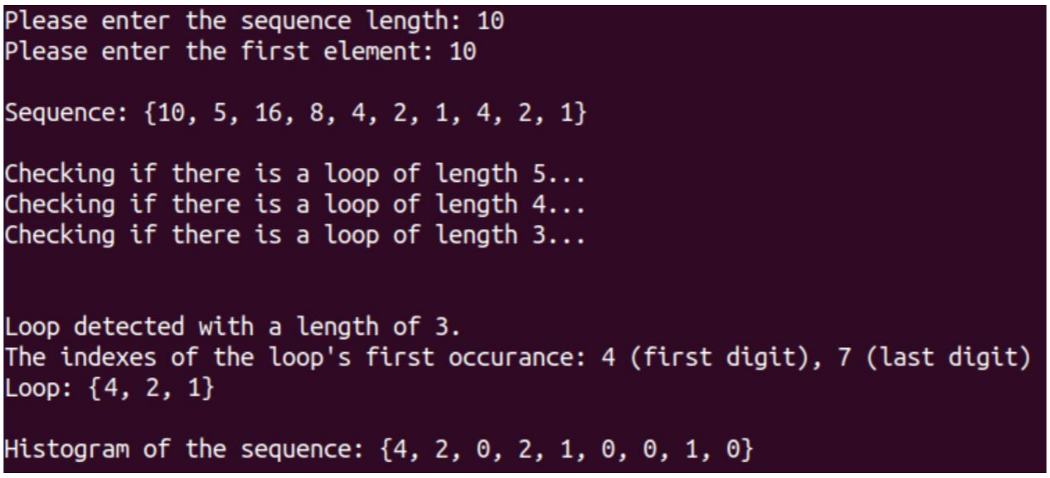


```
Please enter the sequence length: 7
Please enter the first element: 55

Sequence: {55, 166, 83, 250, 125, 376, 188}

Checking if there is a loop of length 3...
Checking if there is a loop of length 2...
No loop found.

Histogram of the sequence: {3, 1, 1, 0, 1, 0, 0, 1, 0}
```



```
Please enter the sequence length: 10
Please enter the first element: 10

Sequence: {10, 5, 16, 8, 4, 2, 1, 4, 2, 1}

Checking if there is a loop of length 5...
Checking if there is a loop of length 4...
Checking if there is a loop of length 3...


Loop detected with a length of 3.
The indexes of the loop's first occurance: 4 (first digit), 7 (last digit)
Loop: {4, 2, 1}

Histogram of the sequence: {4, 2, 0, 2, 1, 0, 0, 1, 0}
```