Midterm 2 Report - Group 7

Q1

To run the program, we can do:

```
$ make q1
$ ./q1
```

It will create a text file named “q1-{yourlocaltime}.txt” and start to log the value into it.

The value of tm_sec increases slowly as the program runs every 2~4 minutes. The reason is that the sleep() command registers a callback event to OS, but OS doesn’t guarantee to wake the process up at an accurate time due to CPU scheduling or heavy system load.
For a program like the `cron` daemon, it may sleep for less time every minute (e.g. 55 seconds) and wake up earlier to synchronize and eliminate the deviation of seconds. One possible solution is to sleep for less time by checking system factors that could require a change in sleep timing.

Q2

To run the program, we can do:

```
$ make q2
$ ./q2 ./path-to-directory
```

To test correctly, one may want to use a directory with symbolic links, or create one in the directory.

```
$ ln -s Makefile makelink
$ ./q2
You must provide exactly one argument for the given path
$ ./q2 .
Symbolic link: ./makelink -> Makefile
```

Q3

To run the program, we can do:

```
$ make q3
$ ./q3
13:54:02, Wednesday December 06, 2023
```

To compare with current time, we can use date with the same format string:

```
$ date +"%H:%M:%S, %A %B %d, %Y"
13:54:04, Wednesday December 06, 2023
```

Q4

To run the program, we can do:

```
$ make q4
$ ./q4
```

The lines 55 and 61 contain the following variable declaration:

```
struct ListNode newNode;
```

This variable since it is initialized inside the function is allocated in the stack frame for the function. Hence, once the function returns, the value of newNode is no longer available to the pointers that use its address on the stack.

Hence, we must change the function to return a valid address using heap allocation. We use malloc here to make sure that the variable data remains persistent after the function's execution:

```
struct ListNode *newNode = (struct ListNode
*)malloc(sizeof(struct ListNode));
```

The rest of the statements then have to be modified to use the new pointer for lined list logic:

```
     if(head == NULL){
      struct ListNode *newNode = (struct ListNode
*)malloc(sizeof(struct ListNode)); // immediate variable,
needs a malloc call to be allocated on heap
      newNode->val = val;
      newNode->next = NULL;
      head = newNode;
```

```
      return head;
    }else {
      struct ListNode *newNode = (struct ListNode
*)malloc(sizeof(struct ListNode)); // same as above, requires
a malloc call for heap allocation
      struct ListNode *tail = head;
      while(tail->next != NULL)tail = tail->next;
      newNode->val = val;
      tail->next = newNode;
      newNode->next = NULL;
    }
    return head;
```

The code then runs correctly:

```
$ ./q4
1
2
3
4
5
6
```