Midterm 2 Programming Report

Notice: Use GNU make (gmake) to compile source codes simultaneously.

Q1

1. How to Run
   ./q1

2. Design
   a. Initialize a counter to count to 6
   b. Fetch the current time and store it to variable *tv*
   c. Initialize a *tm* variable to store the localtime
   d. Enter the infinite loop, each iteration should increment the counter and sleep for 10 seconds. If the counter reaches 6, we reset the counter, fetch the local time, and print the second.

3. Output for 3 hours
   It repeatedly outputs the same second as when we run our program. The reason is because it updates every minutes, and every minutes, when it fetch the local time, the second will always be the same, even for 3+ hours

4. Comparison to cron daemon
   In this case, cron daemon program's way of handling this situation is more efficient and accurate, since we can directly specify at which minute, hour, day, month, week our task should run.
   Our program will print the seconds field every minute, but due to the nature of sleeping for 10 seconds between iterations, there might be a slight drift in the execution time compared to a perfectly synchronized execution like that of a cron job running every minute.

Q2

1. How to Run
   ./q2 ./path/to/search/for/symbolic/links

2. Design
   a. Check if the argument is correct, if not output instructions on how to use.
   b. We make use of the instructions available to the terminal, which is find function. First, we store our instructions on a variable *buffer*.
   c. The instructions will be :

   ```
   find <path> -type l -exec ls -la {} \\;
   ```

   Breaking down the command, we use the **find** function to search for symbolic links **-type l** under the specified path of **<path>,** which would be a user input. For each symbolic link found, it executes the ***ls -la*** command to list detailed information about the symbolic link. The ***{}*** in the command is a placeholder that gets replaced by the current file or directory being processed by ***find*** when executing the ***-exec*** command. The \\ at the end is used to terminate the ***-exec*** command for ***find***.
   d. We input the buffer to the system by using ***system(buffer)*** function

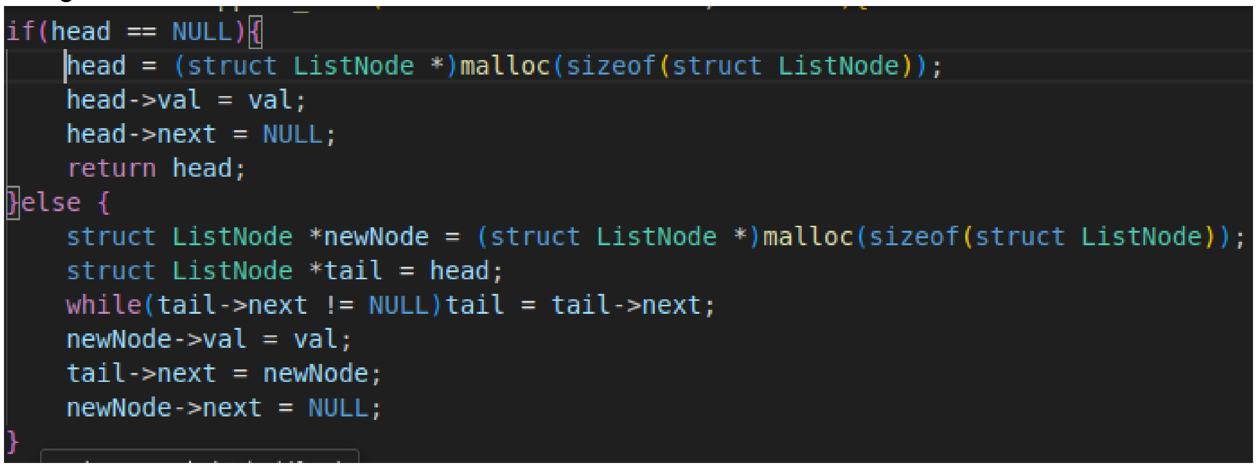
Q3

1. How to Run
   ./q3

2. Design
   Step 1: Use time(2) to get the current time for Epoch in seconds
   Step 2: Use localtime_r() to break down “time_t” and stores in the tm structure
   Step 3: Use strftime(3) to format the output buffer. %X stands for the time(24 hour format), %A stands for full weekday name, %B stands for full month name, %d stands for the day of the month, and %G stands for ISO 8601 year.
   Step 4: print the buffer out.

Q4

1. How to Run
   ./q4
2. Design



```
if(head == NULL){
    head = (struct ListNode *)malloc(sizeof(struct ListNode));
    head->val = val;
    head->next = NULL;
    return head;
}else {
    struct ListNode *newNode = (struct ListNode *)malloc(sizeof(struct ListNode));
    struct ListNode *tail = head;
    while(tail->next != NULL)tail = tail->next;
    newNode->val = val;
    tail->next = newNode;
    newNode->next = NULL;
}
```

(Segmentation fault) First, We need to deal with the fact that we don't allocate memory to ListNode. When we want to add a ListNode and assign value to it, we need to allocate memory with malloc(3) to it.
(Warning) Second, We remove newNode when head == NULL to avoid returning a local variable in the function. Instead, We directly assign val and NULL to head->val and head->next respectively, and then return head.