CS220 - Computer System II
Combined Assignment 8 and 9 (200 points)

Due: 11/04/2016, 11:59pm

1 Instructions

- Answer the questions individually. Group effort is not allowed.
- Reading: `http://c-faq.com`, K&R C, chapter 1-6.

2 Questions

1. (100 points) You are given structure Node below, where next points to the next node in the list, and prev presents to the previous node in the list (Figure 1):

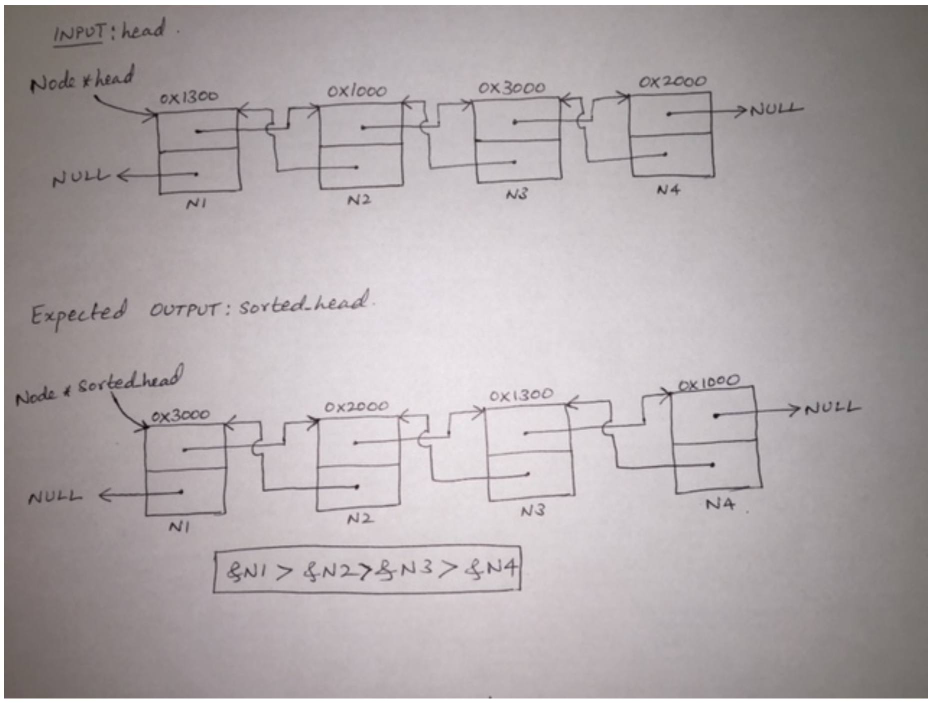
```
struct _Node {
  struct _Node *next;
  struct _Node *prev;
};
typedef struct _Node Node;
```

Implement a function `Node *sort_nodes(Node *head)` that returns head of list such that:
$\&node1 > \&n2 > \&n3 > ... > \&last\_node$

NOTE:

- For this assignment, you will not be given driver code. You will have to write your own code to generate lists and test your sort_nodes function. You can mix nodes from stack (local variables), heap (dynamic variables) and global variables to generate a list comprising of nodes of varying addresses.
- Although you are free to use any sorting algorithm, it might be easier to use selection sort. In simple words, selection sort is where you iterate through the list to select the largest element and move it to the beginning of the list. (https://en.wikipedia.org/wiki/Selection_sort).
- You are guaranteed that your function will be tested against lists that contain at least 3 elements. So, you are not required to handle lists of smaller size.
- You are not allowed to make copies of the nodes. You are required to sort them by altering the next/prev pointers.

Figure 1: Lists before and after sorting. Arrows going left to right indicate “next” pointers, and arrows going right to left indicate the “prev” pointers.



- You will submit sorter.c and node.h. sorter.c will contain the sort_nodes function and node.h will contain the definition of the Node structure and the prototype for the sort function.

2. (80 points) You are to write a program to estimate the throughput of nop instructions. That is, you are to find the number of nop instructions that the processor can execute in 1 second. Your program must be called "estimator". It should print a single number indicating the number of nop instructions the processor can execute in 1 second.

   HINT: Using the timer code in Lab8, run an assembly function comprising of 100 nop instruction a large number of times (e.g., (unsigned int) 0xffffffff), and run another assembly function that contains 0 nop instructions (only a ret instruction) the same number of times. In both cases, measure the time consumed over 5-10 trials. Compute the average number of nop instructions the processor executes per second. You can expect at least 2 x 10^9 nop's per second.

3 Submitting your code

(20 points) When you are ready to submit your code, store sorter.c, node.h, estimator.c, estimator.S and a Makefile in a folder named "Assn8". Upon running "make", your Makefile should generate libsort.so, a dynamic library, and libsort.a, a static library of the sort_nodes function. It must also generate an executable named estimator. Compress Assn8 folder to obtain assn8.tar.gz. Submit assn8.tar.gz on Blackboard.