Ankara University
Computer Engineering
COM2067 LAB 2

In this lab, you are expected to write a function that finds intersecting elements in linked lists received from the user. The function prototype is given below. Your program will read 2 linked lists (read the elements) from the user. By entering a value of -1, the user will terminate the process of providing elements for the linked list. These linked lists will be passed to the function and the intersecting elements will be returned to the main function as a new linked list. The main function will also print this new linked list. If there is no intersecting element, print “-1” on the screen.

```
struct node {
    int data;                   // value of element in linked list
    struct node* next;          // pointer to point to the next element in the linked list
};

struct intersectedNode {
    int data;                   // value of element in linked list
    int index1;                 // the index of the intersecting element in the first linked list
    int index2;                 // the index of the intersecting element in the second linked list
    struct node* next;          // pointer to point to the next element in the linked list
};
```

struct intersectedNode* intersected_nodes(struct node* list1, struct node* list2)

Programs that do not use the function defined above or change their prototype will not be evaluated.

Example:

List1: 27 -> 3 -> 25 -> 7 -> 21 -> 11 -> 13 -> 19 -> 17 -> 15 -> 9 -> 23 -> 5 -> 1

List2: 2 -> 5 -> 23 - > 19 -> 11 -> 13 -> 7 -> 3

New List: 3 -> 7- > 11 -> 13 -> 19 -> 23 -> 5

Submission:

Name your source file as <StudentID>.c. For example, if your ID is 22290777, then you will submit 22290777.c file.

Testing:

We provide a sample input/output text file pairs for you to test your codes at Ubuntu. Please carefully review the sample input and output files given to you for the correct output format.

We recommend you to use input redirection mechanism of your operating system to test your programs. For example, if your executable is called as Lab2, redirect the input.txt file to standard input using < operator and redirect your outputs to a file using > operator such as:

```
> ./Lab2 < input.txt > output.txt
```

This kind of execution enables your programs to read inputs from a file without writing any file related functions. In other words, scanf reads data from the redirected files instead of the std. input in this way (e.g. keyboard).

Automatically compare your own output with the expected output by using the diff myOutput1.txt output1.txt command. If a warning as shown below does not appear on the screen after executing this command, this means that your program is working correctly. If you see a warning in the command system after executing the command, this indicates that there is a problem with your output.

Test your program for different inputs that you will create yourself. Please note that the input files given to you and the input files used during the evaluation may differ from each other.