The following problems create and use singly linked lists. You can use the following structure definition for nodes.

```
struct node {
 int value;
 struct node *next;
};
```

Use linked list. Using arrays will not receive any points.

Problem 1

Write a program in C to create a singly linked list of `n` nodes and display the nodes in original and reversed orders. Your program should first ask user for integers. In a loop, it reads in integers (`scanf`), creates structure nodes (`malloc`), saves the integers into the nodes, and links the nodes onto a ***linked list***. To finish inputting integers, the user presses ctrl-d, and scanf() will return EOF (e.g., `if(scanf("%d", ...) == EOF) break;`)

Your program prints the integers from the beginning of the list to the end, and then it prints the integers in reverse order (i.e., from the end to the beginning).

Hint: To print the integers in reverse order, you can create a new linked list: remove the nodes on the original list from beginning to end, and add then to the front of the new list. In this way, the last node in the original list will become the first node in the new list, the second last becomes the second, … Your program can display the nodes on the new list. Do not use an array to save the data in the original list, and then display the data in the array in reversed order.

Test Data :

Input data for node 1 : 5
Input data for node 2 : 6
Input data for node 3 : 7

Expected Output:

Data entered in the list are:
Data = 5
Data = 6
Data = 7

The list in reverse are :
Data = 7
Data = 6
Data = 5

Problem 2

Write a C program that use **bubble sort** to sort the ***integer values*** provided through user inputs (***not from the command line***). Your program should first ask user for integers. In a loop, it reads in integers (`scanf`), creates structure nodes (`malloc`), saves the integers into the nodes, and links the nodes onto a ***linked list***.

To finish inputting integers, the user presses ctrl-d, and scanf() will return EOF. Then, your program sorts the nodes on the list by moving the nodes on the list. The nodes saving smaller values are moved to the front and the nodes saving larger values are moved to the rear.

Use bubble sort. Do not use radix sort or other sort algorithms. Sort the nodes on the linked list directly. Do not use an array to save and sort the values. I know you can swap the values in the nodes to keep them in ascending order. But this problem is for you to practice. Thus, your program is not allowed to change the value in a node after the value is initialized. To sort the values, your program can only change the positions of the nodes on the list (i.e., unlink a node from the list and relink it to somewhere else).

When sorting is finished, your program prints out the integers saved in the nodes from the first node to the last node on the list, **one number on each line**.

Test your program manually first. Manually type in inputs, press ctrl-d, and check output. To fully test your program with more numbers, modify and use the following scripts. $1 of the script is the count of the integer values. Take screenshots when you test your program manually, such that we can see the numbers provided to the program and the output of the program.

```
#!/bin/bash
count=$1
rm input
rm your_output
rm standard_output
echo "================== input ======================="
for (( i=0; i<${count}; i++ ));
do
 echo $RANDOM
done | tee -a input
echo "============= execution result ================"
cat input | PATH_NAME_OF_YOUR_PROGRAM | tee your_output
sort -n input > standard_output
echo "====== differences from correct result ======="
diff your_output standard_output
```