Solution for 2024-2025 DSA Exam, first sitting, Code 2, solution for max 6 points.

This problem has a requirement for maximum of 6 points and a requirement for maximum 10 points. Only one of these requirements will need to be provided with a solution, the choice is yours which one you wish to attempt.

The first line of your file will contain a one line comment with either 6 or 10 written. This will be the basis on which the code will be tested. Failure to follow this will lead to the impossibility of testing your code which will result in minimum marks.

Write, in C, the necessary data structures and algorithms which do the following:

- Create a dynamically allocated linked list of nodes with 32 bit unsigned integers as data payload based on whitespace separated values from a file. The name of the file, which contains full relevant path, will be the only command line argument for your compiled program. The size of this file is not considered known.
- Sort this list so when traversed from the beginning to end it will generate an ordered sequence of those 32 bit unsigned integers, without touching the same node more than once.
- Traverse this list only one time and print the values of each node to the standard output. Values will be separated by one space. The last value will also have one space after it. This will output an ordered sequences for those values.

Constraints:

- Any form of array is off limits.
- Unnecessary code will result in points deductions.
- No unnecessary memory allocations. After the file is read and the list generated, no more memory allocations should happen.
- Memory leaks will trigger massive points deduction.
- Make sure to free the memory before exit.
- Code should be neatly divided into relevant functions.
- Code should be legible. If you think the grader will need more than a few minutes to understand what you did, please explain what that section of code does using comments
- There is a maximum of 20 seconds execution time when testing. If the code takes longer then it is considered that it does not work at all.

Constraints for the maximum 6 marks solution:

- There are no other constraints

Constraints for the maximum 10 marks solution:

- The list must be built in the order the values are read from the file
- After the list is built, the payload data of a node should not be changed
- After the list is built then its nodes can be sorted (i.e. do not sort the list while building it)

```c
#include<stdio.h>
#include<stdlib.h>

//---------------------------------------
typedef struct nodeStruct{
        int payload;
        struct nodeStruct *next;
}Node;
//---------------------------------------
Node* freeList(Node *head){
        Node *tmp = head;
        while(head){
                head = head->next;
                free(tmp);
                tmp=head;
        }
return NULL;
}
//---------------------------------------

void printList(Node *head){
        while(head){
                printf("%d ", head->payload);
                head = head->next;
        }
        printf("\n");
}
//---------------------------------------
```

Data structure and type definition

Free the memory to avoid meory leaks.

Don’t forget to return NULL on success so the head pointer outside the function will not be left dangling.

Print the list to stdout

```
27 Node* insertOrderedNode(Node *head, int val){
28         Node* new = (Node*)malloc(sizeof(Node));
29         new->payload = val;
30         new->next = NULL;
31
32         if(!head) return new;
33
34         Node *tmp = head;
35
36         if(tmp->payload > new->payload){
37                 new->next = tmp;
38                 return new;
39         }
40
41         while(tmp){
42                 if(!tmp->next){
43                         tmp->next = new;
44                         break;
45                 }
46
47                 if(tmp->next->payload > new->payload){
48                         new->next= tmp->next;
49                         tmp->next = new;
50                         break;
51                 }
52                 tmp = tmp->next;
53         }
54
55 return head;
56 }
57
```

Allocate memory for the new node.
Don't forget to initialise members.

If list is empty, new node will be head of list.

If there is at least one node in the list and the first value is greater than the new value, the new value becomes the head of the list.

While there are nodes in the list

If the last node is reached and could not find the place for the new value, then the new value will be the (new) last in list.

If we are not on the last node, but the next nodes value is greater than the new value then the new value will be inserted between the current and the next node

```c
Node* listFromFile(char *filepath){
        FILE *f =fopen(filepath, "r");
        Node *head = NULL;

        while(!feof(f)){
                int val = 0;
                fscanf(f, "%d", &val);
                if(feof(f)) break;
                head = insertOrderedNode(head, val);
        }

        fclose(f);

return head;
}
```

Open file / don't forget to close at the end.

Initialise list head.
Make a habit from disliking uninitialised variables, pointers in particullar.

While not the end of file, read value from file and call function to add that value to the list.

```c
int main(int argc, char **argv){
        char *filepath = argv[1];
        Node *head = NULL;
        head = listFromFile(filepath);
        printList(head);
        head = freeList(head);

return 0;
}
```

This is the solution for max 6 points, so we can order the list while populating it.

Main function. Command line arguments are considered correct, so no error checking.
File is considered to exist and be correct so no error checking.

After freeList, **head** will be NULL

We will apply the InsertSort arlgorithm to a dynamically allocated single linked list, therefore sorting the list as each value is inserted into the list.