Sabancı University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Spring 2021

Homework 2 – Displaying Sorted Subsequences with Linked Lists

Due: 17/3/2021, Wednesday, 21:00

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You can NOT collaborate with your friends and discuss solutions. You have to write down the code on your own. Plagiarism and homework trading will not be tolerated!

Introduction

In this homework, you are asked to implement a program that stores and displays subsequences of some input integers in ascending or descending sorted manner. The program must use a *Linked List* structure for storage and processing. The sorting mode (ascending or descending) and the numbers are going to be taken from the user as keyboard input. The program details will be explained in the subsequent sections.

The Data Structure to be Used

In this homework, you **must** use a *linked list* (regular one-way linked list) as your main data structure. The *node* struct of this list must have the following data members (if you want, you can add constructors to the struct).

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

In this node structure, `value` keeps the integer value stored in this node. You can see the visualization of this data structure in Figure 1.

You are not allowed to use arrays, vectors and similar containers (including files) in this homework; all data must be stored and processed within the linked list.

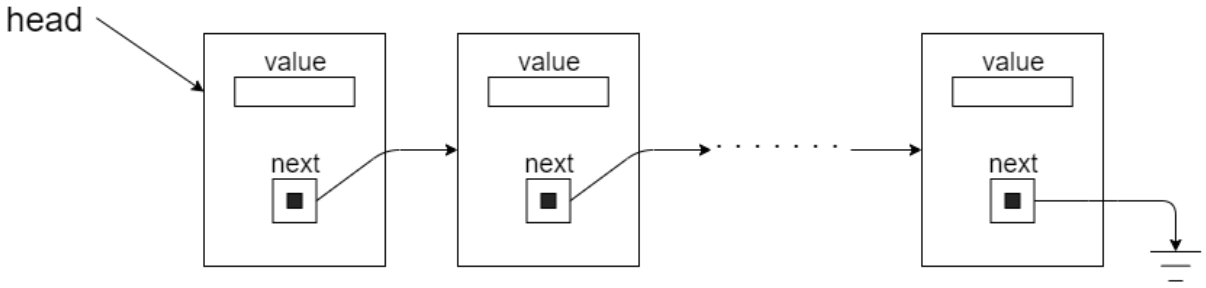


Figure 1: Visualization of Linked List structure

The Program Flow, Inputs and Outputs

Your program is going to start with getting an input from the user regarding the order mode of the list to be constructed. This first input can be `'A'` or `'D'`, which mean *ascending* or *descending*, and it is case sensitive (i.e. uppercase / lowercase difference matters). You should apply an input check here to make sure that the order mode entered is either `'A'` or `'D'`. If not, your program should give an error message and ask for the order mode again until a correct input is entered. Note that, the user can enter any string for the order mode (i.e. not necessarily a single char). In case of entering multiple characters string input for order mode, you should give an error message and ask for new input (see sample runs). On the other hand, if there are multiple strings read in the same line, this is not source of an error per se; in anyway, you have to check the first string of the line and make the input check as mentioned above. However, in any case, you may want to empty the input buffer after reading the order mode using `cin` (correctly or incorrectly). To do so, use `cin.ignore(numeric_limits<streamsize>::max(),'\n');` [1] See sample runs (especially sample run 2) for different cases.

After getting the order mode, your program should ask for the second input: a line with integer numbers. You are expected to read the entire line in a string variable using `getline(cin, numbers)`, where *`numbers`* is a string variable. Before using `getline(cin, numbers)`, your program should empty input buffer, which might not be empty due to previous input, with command `cin.ignore(numeric_limits<streamsize>::max(),'\n');` You can assume that the numbers are entered as integers separated by blank(s) or tab(s), so no input checks are required for the numbers. However, pressing enter without entering any numbers is possible and you have to consider this case by just finishing the program (see the last sample run). After successfully taking inputs from the user, your program should process the numbers one by one (you may use input string streams, `istringstream`, for this purpose).

For each number, the rules of adding it to the linked list are as follows:

- If the number already exists in the list, ignore it and do not change the list.
- If the number does not exist in the list, delete all the nodes with values greater than it (if order mode is ascending) or smaller than it (if order mode is descending). Then, add the number as the last node of the list. Here, be aware of the cases where you may need to change the head of the list.

[1] Basically, this `ignore` function is used to get rid of everything remained in the line that you previously read using `cin`. Sometimes you think nothing remained, but even the end of line character would cause a problem while reading the next input. On the other hand, usage of `ignore` statement might not be needed if you do not use `cin`, and use `getline`, to read the order mode. The reason is that `getline` reads the entire line, including the end of line character, into a string, and nothing remains in the input buffer.

Two processing examples are shown in Figures 2 and 3. For more examples see sample runs.

After processing each number, if the number is already in the list, display a message specifying this and the current list content. If the number is not in the list, you have to display the content of the linked list to the screen together with deleted node values and the appended one. See the sample runs for the output details.

At the end of the program, do not forget to delete the remaining nodes in the lists with an appropriate message.

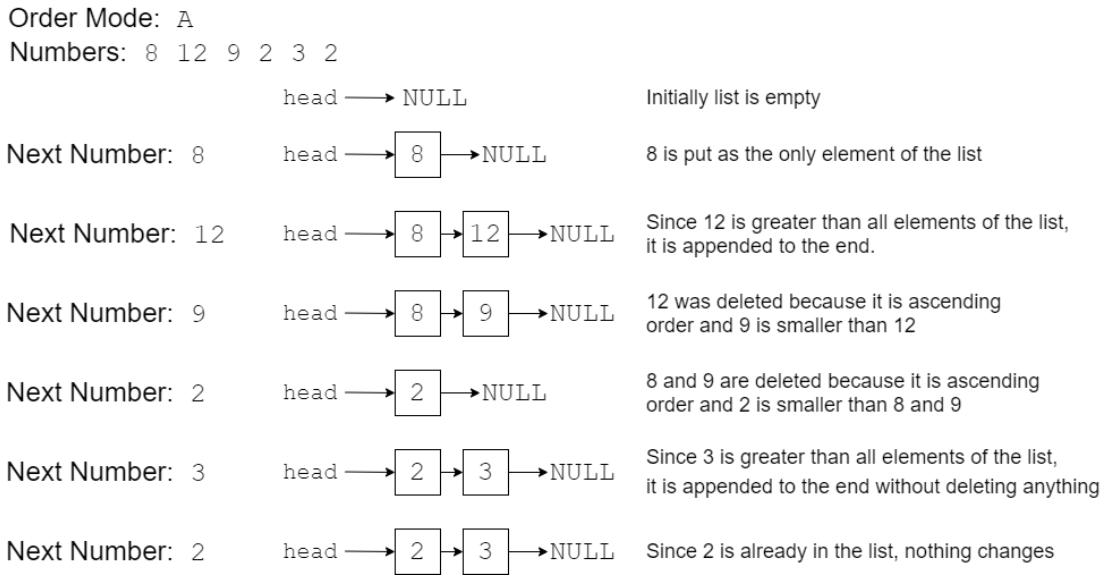


Figure 2: Example input and the corresponding linked lists (ascending order case)

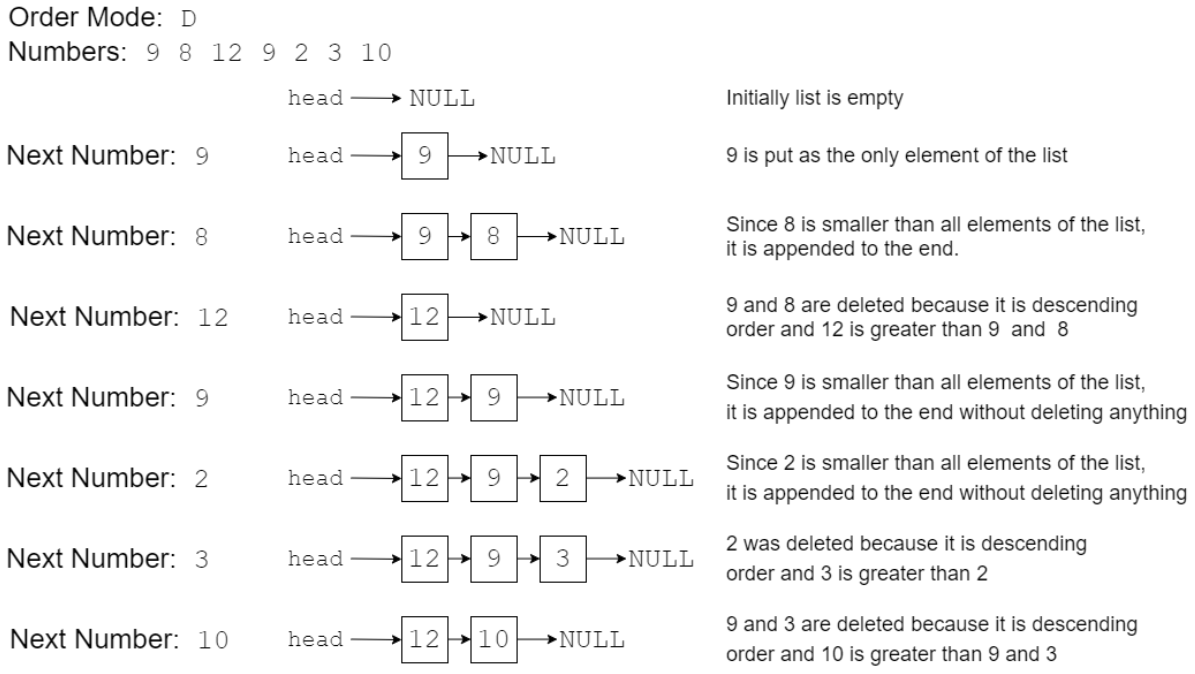


Figure 3: Example input and the corresponding linked lists (descending order case)

Some Important Rules

In order to get a full credit, your programs must be efficient and well presented, presence of any redundant computation or bad indentation, or missing, irrelevant comments are going to decrease your grades. You also have to use understandable identifier names, informative introduction and prompts. Modularity is also important; you have to use functions wherever needed and appropriate.

Since you will use dynamic memory allocation in this homework, it is very crucial to properly manage the allocated area and return the deleted parts to the heap whenever appropriate. Inefficient use of memory may reduce your grade.

When we grade your homework we pay attention to these issues. Moreover, in order to observe the real performance of your codes, we may run your programs in *Release* mode and **we may test your programs with very large test cases**. Of course, your program should work in *Debug* mode as well.

Please do not use any non-ASCII characters (Turkish or other) in your code.

You are allowed to use sample codes shared with the class by the instructor and TAs. However, you cannot start with an existing .cpp or .h file directly and update it; you have to start with an empty file. Only the necessary parts of the shared code files can be used and these parts must be clearly marked in your homework by putting comments like the following. Even if you take a piece of code and update it slightly, you have to put a similar marking (by adding "`and updated`" to the comments below.

```
/* Begin: code taken from ptrfunc.cpp */

…

/* End: code taken from ptrfunc.cpp */
```

Sample Runs

Sample runs are given below, but these are not comprehensive, therefore you have to consider **all possible cases** to get full mark.

Inputs are shown in **bold** and *italic*.

Sample Run 1:

```
Please enter the order mode (A/D): a
Please enter the order mode again (A/D): 123
Please enter the order mode again (A/D): CS204
Please enter the order mode again (A/D): A
Please enter the numbers in a line: 8 12 9 2 6 3 4 3

Next number: 8
Deleted nodes: None
Appended: 8
List content: 8
```

```
Next number: 12
Deleted nodes: None
Appended: 12
List content: 8 12

Next number: 9
Deleted nodes: 12
Appended: 9
List content: 8 9

Next number: 2
Deleted nodes: 8 9
Appended: 2
List content: 2

Next number: 6
Deleted nodes: None
Appended: 6
List content: 2 6

Next number: 3
Deleted nodes: 6
Appended: 3
List content: 2 3

Next number: 4
Deleted nodes: None
Appended: 4
List content: 2 3 4

Next number: 3
3 is already in the list!
List content: 2 3 4

All the nodes are deleted at the end of the program: 2 3 4
```

Sample Run 2:

```
Please enter the order mode (A/D): De cs204 qwerty
Please enter the order mode again (A/D): DE CS204 QWERTY
Please enter the order mode again (A/D): sdfsdf sdahssdf D
Please enter the order mode again (A/D): D 1 4 55 22323
Please enter the numbers in a line: 8 12 9 2 6 3 4 3

Next number: 8
Deleted nodes: None
Appended: 8
List content: 8

Next number: 12
```

```
Deleted nodes: 8
Appended: 12
List content: 12

Next number: 9
Deleted nodes: None
Appended: 9
List content: 12 9

Next number: 2
Deleted nodes: None
Appended: 2
List content: 12 9 2

Next number: 6
Deleted nodes: 2
Appended: 6
List content: 12 9 6

Next number: 3
Deleted nodes: None
Appended: 3
List content: 12 9 6 3

Next number: 4
Deleted nodes: 3
Appended: 4
List content: 12 9 6 4

Next number: 3
Deleted nodes: None
Appended: 3
List content: 12 9 6 4 3

All the nodes are deleted at the end of the program: 12 9 6 4 3
```

Sample Run 3:

```
Please enter the order mode (A/D): A
Please enter the numbers in a line: 10 15 20 25 1 3 8 2 -2 -1 0

Next number: 10
Deleted nodes: None
Appended: 10
List content: 10

Next number: 15
Deleted nodes: None
Appended: 15
List content: 10 15
```

```
Next number: 20
Deleted nodes: None
Appended: 20
List content: 10 15 20

Next number: 25
Deleted nodes: None
Appended: 25
List content: 10 15 20 25

Next number: 1
Deleted nodes: 10 15 20 25
Appended: 1
List content: 1

Next number: 3
Deleted nodes: None
Appended: 3
List content: 1 3

Next number: 8
Deleted nodes: None
Appended: 8
List content: 1 3 8

Next number: 2
Deleted nodes: 3 8
Appended: 2
List content: 1 2

Next number: -2
Deleted nodes: 1 2
Appended: -2
List content: -2

Next number: -1
Deleted nodes: None
Appended: -1
List content: -2 -1

Next number: 0
Deleted nodes: None
Appended: 0
List content: -2 -1 0

All the nodes are deleted at the end of the program: -2 -1 0
```

Sample Run 4:

```
Please enter the order mode (A/D): A
Please enter the numbers in a line: 8 10 11 10 9 10

Next number: 8
Deleted nodes: None
Appended: 8
List content: 8

Next number: 10
Deleted nodes: None
Appended: 10
List content: 8 10

Next number: 11
Deleted nodes: None
Appended: 11
List content: 8 10 11

Next number: 10
10 is already in the list!
List content: 8 10 11

Next number: 9
Deleted nodes: 10 11
Appended: 9
List content: 8 9

Next number: 10
Deleted nodes: None
Appended: 10
List content: 8 9 10

All the nodes are deleted at the end of the program: 8 9 10
```

Sample Run 5:

```
Please enter the order mode (A/D): D
Please enter the numbers in a line: 5 4 3 2 1 0 -1 100

Next number: 5
Deleted nodes: None
Appended: 5
List content: 5

Next number: 4
Deleted nodes: None
Appended: 4
List content: 5 4
```

```
Next number: 3
Deleted nodes: None
Appended: 3
List content: 5 4 3

Next number: 2
Deleted nodes: None
Appended: 2
List content: 5 4 3 2

Next number: 1
Deleted nodes: None
Appended: 1
List content: 5 4 3 2 1

Next number: 0
Deleted nodes: None
Appended: 0
List content: 5 4 3 2 1 0

Next number: -1
Deleted nodes: None
Appended: -1
List content: 5 4 3 2 1 0 -1

Next number: 100
Deleted nodes: 5 4 3 2 1 0 -1
Appended: 100
List content: 100

All the nodes are deleted at the end of the program: 100
```

Sample Run 6:

```
Please enter the order mode (A/D): D
Please enter the numbers in a line: 11       7   5    3      2

Next number: 11
Deleted nodes: None
Appended: 11
List content: 11

Next number: 7
Deleted nodes: None
Appended: 7
List content: 11 7

Next number: 5
Deleted nodes: None
```

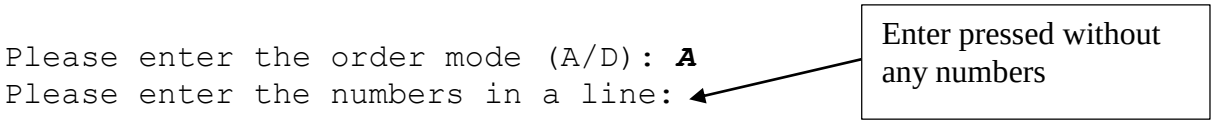
```
Appended: 5
List content: 11 7 5

Next number: 3
Deleted nodes: None
Appended: 3
List content: 11 7 5 3

Next number: 2
Deleted nodes: None
Appended: 2
List content: 11 7 5 3 2

All the nodes are deleted at the end of the program: 11 7 5 3 2
```

Sample Run 7 (empty input for numbers):




```
The list is empty at the end of the program and nothing is deleted
```

What and where to submit (PLEASE READ, IMPORTANT)

You should prepare (or at least test) your program using MS Visual Studio 2012 C++. We will use the standard C++ compiler and libraries of the abovementioned platform while testing your homework. It'd be a good idea to write your name and last name in the program (as a comment line of course).

Submissions guidelines are below. Some parts of the grading process might be automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.

Name your cpp file that contains your main program using the following convention:

"SUCourseUserName_YourLastname_YourNames_HWnumber.cpp"

Your SUCourse user name is your SUNet user name which is used for checking sabanciuniv e-mails (not the numeric one). Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the file name must be:

Cago_Ozbugsizkodyazaroglu_Caglayan_hw1.cpp

In some homework assignments, you may need to have more than one .cpp or .h files to submit. In this case, add informative phrases after the hw number. However, do not add any other character or phrase to the file names. Sometimes, you may want to use some user defined libraries (such as strutils of Tapestry); in such cases, you have to provide the necessary .cpp

and .h files of them as well. If you use standard C++ libraries, you do not need to provide extra files for them.

These source files are the ones that you are going to submit as your homework. However, even if you have a single file to submit, you have to compress it using ZIP format. To do so, first create a folder that follows the abovementioned naming convention ("SUCourseUserName_YourLastname_YourNames_HWnumber"). Then, copy your source file(s) there. And finally compress this folder using WINZIP or WINRAR programs (or another mechanism). Please use "zip" compression. "rar" or another compression mechanism is NOT allowed. Our homework processing system works only with zip files. Therefore, make sure that the resulting compressed file has a zip extension. Check that your compressed file opens up correctly and it contains all of the files that belong to the latest version of your homework.

You will receive zero if your compressed zip file does not expand or it does not contain the correct files. The naming convention of the zip file is the same. The name of the zip file should be as follows:

SUCourseUserName_YourLastname_YourNames_HWnumber.zip

For example, zubzipler_Zipleroglu_Zubeyir_hw2.zip is a valid name, but

Hw2_hoz_HasanOz.zip, HasanOzHoz.zip

are **NOT** valid names.

Submit via SUCourse ONLY! You will receive no credits if you submit by other means (e-mail, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!
Albert Levi, Vedat Peran