

Sabancı University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Spring 2022

Homework 3 – Character Processing using 2D Linked List

Due: 5/4/2022, Tuesday, 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 two-dimensional character processor that will read some words from an input file and store them in a *two-dimensional linked list* structure. Each row in this 2D data structure is a linked list that stores a word character by character; i.e. each node of a row stores one character of the word. Moreover, the characters of the words are also vertically connected via some other pointers, which makes the data structure two-dimensional. Your program will also process and manipulate the data structure via some operations and member functions such as displaying all or certain rows/columns, sorted insertion, deleting certain rows/columns. The program details will be explained in the subsequent sections.

The Program Flow

At the very beginning, the program asks the user for the input file name. The user enters the file name and if the file is not successfully opened, the program keeps reading the file name over and over again until it is opened successfully. This file stores some words (assume all lowercase letters), line by line. Then, the program should read each word of the file and stores in the 2D data structure in ascending alphabetical order with respect to the first letter. After that, the program prompts a menu that contains a list of operations. The menu provides 7 different options, and the options from 1 to 6 can be chosen in any order several times. When one of the options from 1 to 6 is entered, the corresponding operation is performed and then the next menu option is read. This continues until the user chooses the seventh option (*Clear the matrix and exit*) to delete the 2D data structure and end the program. In case the user enters a wrong menu option, an error message

is displayed. Actually, the construction of the data structure by reading the file and the abovementioned menu logic are implemented by us and given to you as a cpp file that contains the main function. In this main function, we call several member functions for the class that you will implement.

The menu options and the corresponding operations are explained below.

1. Display the full Matrix:

When this option is selected, the entire 2D data structure should be displayed row by row, starting with the topmost one.

2. Display the full matrix in reverse order

When this option is selected, the entire matrix should be displayed again row by row but in reverse order, i.e., starting with the bottommost row and going up one by one. The characters in the rows will be displayed in normal order (not reverse). Please remark that this option is for display purposes only; the data structure itself will not change. Hint: you can make use of recursion here.

3. Display all the rows starting with a specific character

When this option is selected, the program will ask the user to enter a character. Then, the program will display all the rows that start with this character.

4. Display all the columns starting with a specific character

When this option is selected, the program will ask the user to enter a character. Then, the program will display all the columns that start with this character.

5. Delete all the rows starting with a specific character

When this option is selected, the program will ask the user to enter a character. Then, all the rows that start with this character will be deleted. This option should also display the total number of rows deleted.

6. Delete all the columns starting with a specific character

When this option is selected, the program will ask the user to enter a character. Then, all the columns that start with this character will be deleted. This option should also display the total number of columns deleted.

7. Clear the data structure and exit

When this option is selected, the program should delete all the dynamic memory allocations for the data structure and exit the program.

The Data Structure to be used

In this homework, you are asked to implement the two-dimensional linked list data structure as a *class* (named `TwoDLinkedList`) with one private data member, which is a pointer to the *head* node of the data structure (the one at the top-left). The node struct of this data structure must have the following data members (if you want, you can add constructors to the struct).

```
struct node
{
    char data;
    node *right;
    node *down;
};
```

The **right** pointer points to the node on the right on a row; whereas, the **down** pointer points to the node at the same position of the row below. The **right** pointer of the last node of each row is NULL. Similarly, the **down** pointer of the bottom node of each column is also NULL. In this way, we generate a matrix shaped 2D linked data structure. In this data structure, the number of nodes in all of the rows are the same. A generic form of such a structure is given in Figure 1. An example data structure with 4 rows and 5 columns is given in Figure 2.

TwoDLinkedList
object

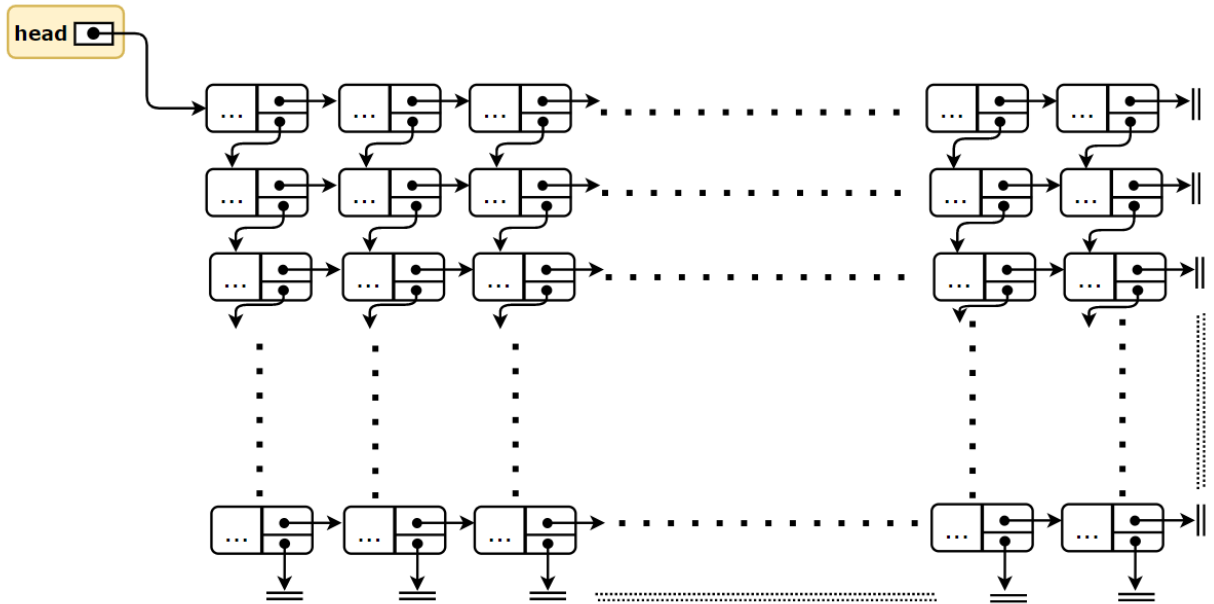


Figure 1. A generic form of the 2D data structure

TwoDLinkedList
object

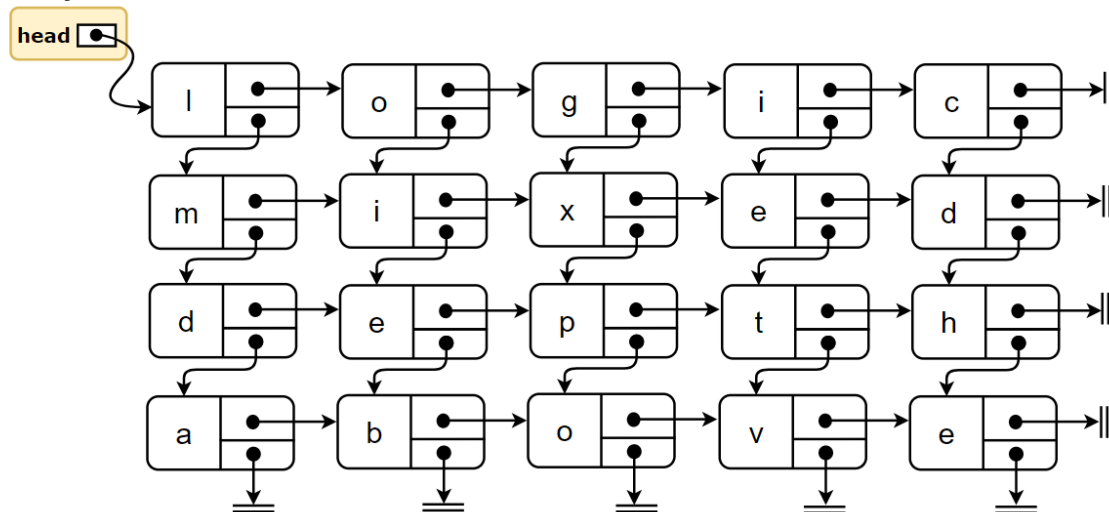


Figure 2. An example 2D data structure with 4 rows and 5 columns

No other multiple data containers (built-in array, dynamic array, vector, matrix, 2D array, etc.) can be used. Do not use string data type to avoid these restrictions. The only places of using string are (i) as file names, and (ii) while reading the words from the file (in the insertion member function, you will use a string parameter to pass the word to the function and process it while creating a row of the 2D data structure).

Reading the File and Creating the Data Structure

While reading the words from the input file, your program should add each word (line) in a separate row of the 2D data structure, where each character is stored in a node. While doing so, your program should maintain ascending order for the rows according to (only) the first character of the row. If there is a tie with the first characters (i.e. having two or more words that start with the same character, you will not check the second characters, etc. If there is such a tie, then the new word should be inserted before the first row that starts with that character. An example file and the resulting 2D structure is given in Figure 3. In this example data structure, "week" is above "weak", because first "weak" is added to the data structure and when "week" is read, it is inserted above "weak" as both start with "w".

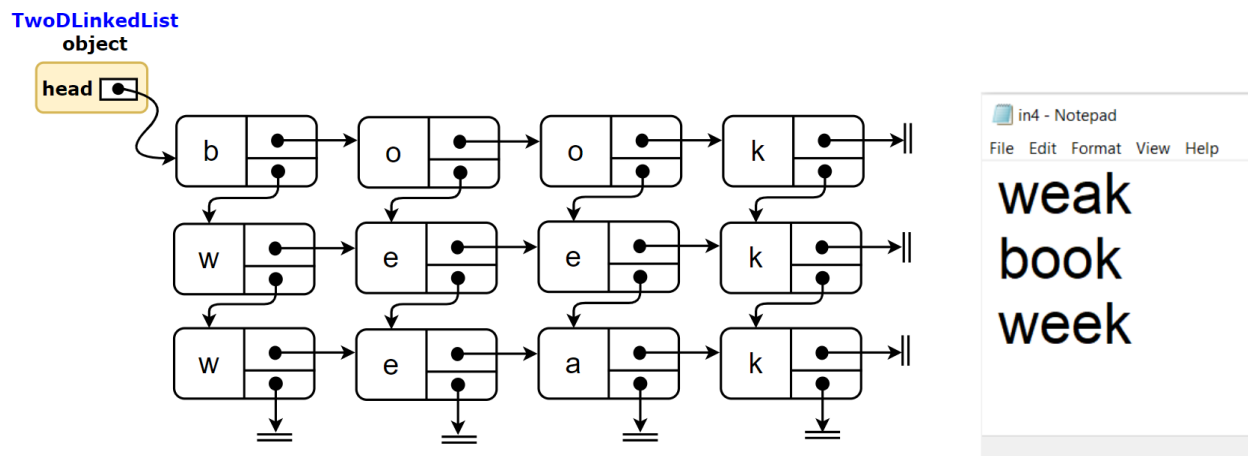


Figure 3. A visualization of the data structure after reading the input from file “in4.txt” and inserting them in the 2D data structure according to the insertion rules.

Note1: For a given input file, we assume that all the lines have the same length. In other words, the input file may have one, two, or more lines; each line has one word and all the words in the same file have the same amount of characters. This is an assumption and you do not need to make an input check for it. With this assumption, we make sure that the data structure created will be a matrix shaped (i.e. same amount of nodes in each row).

Note2: Please assume that the file contains all lowercase letters and there are no whitespaces before, within and after the words. You do not need to check this.

Note3: As mentioned above, you will create a sorted data structure after reading the file. However, since you will make some deletions, the sortedness of the data structure may not be held in the entire program. You do not need to re-sort the data structure after deletions.

Provided main cpp file and the requested class implementation

We have provided the main function of this homework to you within the homework pack. You have to use it directly and **without any modifications** other than adding an `#include` line for your class header file (make sure that the header file name that you `#include` matches the physical file name, if you update the file names to follow the naming convention just before the submission).

As you see from the provided main function, the file opening/reading logic, construction of the data structure via calling a class member function, menu logic are all implemented. What you have to do is to write the class' header and implementation files so that they work with the given main function.

The member functions used in main are self-explanatory and match with the menu operations. The only member function not used in the menu options is `add_row_sorted` function, which takes a string and inserts its characters to the 2D data structure as described in the sections above. We expect you explore the member function prototypes and requirements by analyzing the given main program and the explanations in this homework document.

If you need more member functions, other than the ones that we use in main, you are more than welcome to write. However, please first read the "Object Oriented Design Manifest" part below.

In the main function, after each input we use the command `cin.ignore(numeric_limits<streamsize>::max(), '\n');` so that extra characters in the line are discarded. In this way, for each input line after the menu is displayed, only one (the first) character is read. This eliminates some unexpected behaviors in wrong data entry and makes your program more robust.

In the main, we did not make input check if the characters are digits (for menu) and lowercase letters (for row/column display/delete options). The reason for the latter case is that if the character entered is not a lowercase letter, no row/column can start with it and no display/deletion is done (of course, if the corresponding member functions are implemented properly). For the menu option, although it is a digit, we read as character and if the entered character is not a digit, the code gives an invalid menu option message, as expected.

Object Oriented Design Manifest

We believe it is clear that you have to apply proper object oriented design principles in this homework, but our past experience says that most of the students are either unaware or not willing to follow these principles. Thus, we wanted to manifest some important rules about the class design and implementation.

1) According to the rule #1 of object oriented design, each member function **must be multi-purpose and must perform as single task as possible**. We picked the member functions used in main using this principle. You may add some extra member functions to be used internally for the implementations of the member functions (not in main since you are not allowed to change main), but please do not forget this principle while doing so.

2) Another important rule of object oriented programming is to hide the details of the class from the programmer that uses this class. Particularly for this homework, this means that in any free function and in main function, we should not reach the head of the data structure directly. We have to make all of the updates, searches, etc. in main program through the member functions. Yes, technically it is possible and really easy to write a member function that returns head so that we can freely manipulate the data structure in the main program, but this is totally against the spirit behind object oriented programming and **we do not do this**. Actually, since you are not allowed to change main, we guarantee that you will not manipulate the data structure using head in main, but we wanted to make this clarification here.

3) In a separate cpp file, you have to have the member function implementations. If you want to have some extra free functions to help the implementation of the member functions, also write them in this file (not in main file). If you will write some helper free functions, please remark that the rule #2 above is also valid for free functions.

4) Class and struct definitions will be in a header file.

5) **Yes, you will also submit the main file**; so totally you will submit **three** files (two cpp, one h).

Some Important Rules

In order to get a full credit, your programs must be efficient and well presented, presence of any redundant computation or unnecessary memory usage 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 using classes is mandated in this homework, a proper object-oriented design and implementation will also be considered in grading.

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 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 */
```

```
...
```

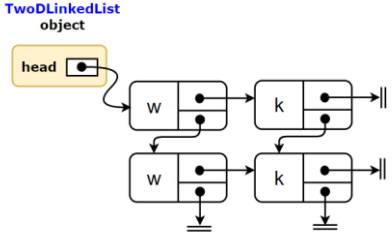
```
/* End: code taken from ptrfunc.cpp */
```

Walk-through Example

A walk-through run of the program is shown in Table 1. The example given in the table assumes that data from “in4.txt” file and has been successfully read and stored in the data structure.

Table 1. Walk through example

Chosen Option (input, if any)	Effect	Output	Explanation
			The figure on the left is a visualization of the data structure storing the input from the “in4.txt” text file. *
1		book week weak	The content of the data structure is displayed on the screen in its 2D representation
2		weak week book	The content of the data structure is reversely displayed on the screen. (You can think of it as flipping the matrix across the x-axis).
3 w		week weak	The program displays all the rows that start with the input char (here, w).
4 o		o e e o e a	The program displays all the columns that start with the input char (here, o).
5 b			The program deletes all the rows that start with the input char (here, b).

1		week week	The content of the data structure is displayed on the screen
6 e			The program deletes all the columns that start with the input char (here, <i>e</i>).
1		wk wk	The content of the data structure is displayed on the screen
7			The data structure is cleared, and the program terminates

* Note that the rows of the matrix are sorted alphabetically according to the first letters. Thus, *book* is at the first row. As there is a tie between *week* and *weak* according to their first letter *w*, the rule that the program followed was to insert the new word (in case of a tie) before the existing one. And as the word *weak* appears first in the file and therefore added first to the data structure, when the word *week* is to be inserted in the matrix, it was inserted before the row of word *weak*.

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*.

The sample input files are provided in the homework pack.

Sample Run 1:

Please enter the file name: ***in9.txt***

Please enter the number of the option from the list below:

1. Display the full matrix
2. Display the full matrix in reverse order
3. Display all the rows starting with a specific character
4. Display all the columns starting with a specific character
5. Delete all the rows starting with a specific character
6. Delete all the columns starting with a specific character
7. Clear the data structure and exit

Please enter a menu option: **1**

aardvarks

Please enter a menu option: **6**

Please enter first character of the columns to delete: **a**

The number of deleted columns is: 3

Please enter a menu option: **1**

rdvrks

Please enter a menu option: **6**

Please enter first character of the columns to delete: **r**

The number of deleted columns is: 2

Please enter a menu option: **1**

dvks

Please enter a menu option: **7**

Clearing the data structure and exiting the program...

Sample Run 2:

Please enter the file name: **in9_2**

Please enter the file name: **in9_2.txt**

Please enter the number of the option from the list below:

1. Display the full matrix
2. Display the full matrix in reverse order
3. Display all the rows starting with a specific character
4. Display all the columns starting with a specific character
5. Delete all the rows starting with a specific character
6. Delete all the columns starting with a specific character
7. Clear the data structure and exit

Please enter a menu option: **1**

macaroons

vacancies

Please enter a menu option: **3**

Please enter first character of the rows to display: **h**

Please enter a menu option: **3**

Please enter first character of the rows to display: **v**

vacancies

Please enter a menu option: **4**

Please enter first character of the columns to display: **o**

o
c

o
i

Please enter a menu option: **4**

Please enter first character of the columns to display: **t**

Please enter a menu option: **4**

Please enter first character of the columns to display: **r**

r
n

Please enter a menu option: **6**
Please enter first character of the columns to delete: **o**
The number of deleted columns is: 2

Please enter a menu option: **5**
Please enter first character of the rows to delete: **m**
The number of deleted rows is: 1

Please enter a menu option: **1**

vacanes

Please enter a menu option: **7**
Clearing the data structure and exiting the program...

Sample Run 3:

Please enter the file name: **in4.txt**

Please enter the number of the option from the list below:
1. Display the full matrix
2. Display the full matrix in reverse order
3. Display all the rows starting with a specific character
4. Display all the columns starting with a specific character
5. Delete all the rows starting with a specific character
6. Delete all the columns starting with a specific character
7. Clear the data structure and exit

Please enter a menu option: **1**

book
week
weak

Please enter a menu option: **6**
Please enter first character of the columns to delete: **j**
The number of deleted columns is: 0

Please enter a menu option: **6**
Please enter first character of the columns to delete: **k**
The number of deleted columns is: 1

Please enter a menu option: **1**

boo
wee
wea

Please enter a menu option: **2**

wea
wee
boo

Please enter a menu option: **5**
Please enter first character of the rows to delete: **d**
The number of deleted rows is: 0

Please enter a menu option: **5**
Please enter first character of the rows to delete: **b**
The number of deleted rows is: 1

Please enter a menu option: **1**

wee
wea

Please enter a menu option: **6**
Please enter first character of the columns to delete: **q**
The number of deleted columns is: 0

Please enter a menu option: **Q**

Invalid input. You need to enter a valid menu option.

Please enter a menu option: **6**
Please enter first character of the columns to delete: **H**
The number of deleted columns is: 0

Please enter a menu option: **6**
Please enter first character of the columns to delete: **w**
The number of deleted columns is: 1

Please enter a menu option: **1**

ee
ea

Please enter a menu option: **7**
Clearing the data structure and exiting the program...

Sample Run 4:

Please enter the file name: **int.txt**
Please enter the file name: **in2.txt**

Please enter the number of the option from the list below:

1. Display the full matrix
2. Display the full matrix in reverse order
3. Display all the rows starting with a specific character
4. Display all the columns starting with a specific character
5. Delete all the rows starting with a specific character
6. Delete all the columns starting with a specific character
7. Clear the data structure and exit

Please enter a menu option: **1**

be
hi
he
is

if
on
pi
we

Please enter a menu option: **3**

Please enter first character of the rows to display: **i**

is
if

Please enter a menu option: **5**

Please enter first character of the rows to delete: **i**

The number of deleted rows is: 2

Please enter a menu option: **1**

be
hi
he
on
pi
we

Please enter a menu option: **4**

Please enter first character of the columns to display: **e**

e
i
e
n
i
e

Please enter a menu option: **6**

Please enter first character of the columns to delete: **e**

The number of deleted columns is: 1

Please enter a menu option: **1**

b
h
h
o
p
w

Please enter a menu option: **6**

Please enter first character of the columns to delete: **b**

The number of deleted columns is: **1**

Please enter a menu option: **1**

Please enter a menu option: **7**

Clearing the data structure and exiting the program...

What and where to submit (PLEASE READ, IMPORTANT) - Same as before except 1st line

You have to submit your class' implementation file, header file and the file that contains the main function.

You should prepare (or at least test) your program using MS Visual Studio C++. We recommend using 2012 version; however, if you use another version provided by Sabancı University software repository, it is OK as long as you specify which version you use at the beginning of your program. 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). Using other platforms (Xcode, VSCode, etc.) is risky and may cause some incompatibility problems.

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 Özbugsizkodyazaroglu, then the file name must be:

Cago_Ozbugsizkodyazaroglu_Caglayan_hw2.cpp

Actually, it does not matter whether you use uppercase of lowercase letters in the file names.

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.

In some homework assignments, you may need to have more than one .cpp or .h files to submit. In this case, use the same filename format but add informative phrases after the hw number (e.g. Cago_Ozbugsizkodyazaroglu_Caglayan_hw2_myfuncs.cpp or Cago_Ozbugsizkodyazaroglu_Caglayan_hw2_myfuncs.h). 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 by using the same naming convention mentioned above. If you use standard C++ libraries, you do not need to provide extra files for them.

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, Ahmed Salem