 [Translated from Hebrew to English - www.onlinedoctranslator.com](https://www.onlinedoctranslator.com/en/?utm_source=onlinedoctranslator&utm_medium=docx&utm_campaign=attribution)

**Working2 – files, structures and dynamic arrays**

**The purpose of the work:**

The work deals with reading and writing files and using dynamic arrays.

The data structures you build will be used by you to answer the queries indicated in the menu.

During the work you will be asked to answer each of the queries posed by the user and to update the corresponding file.

**Job description:**

In this work you have to rate the viewing rates of TV series provided by 2 companiesNetChain and SeakNet.

Each of these companies provide a collection of classic and original TV series.

The owners of the company are interested in ranking the series they own according to the viewing index of its customers.

The customers' viewing data is captured as pairs of data: series code and season number.

For each company a file containing the data of the series is saved, for each series a line is saved with the following data:

* The name of the series – 20 characters at most
* The series code – 3 digitsAt most
* The number of seasons – 2 digitsAt most
* Is the series active - character ('Y' or 'N')

**pay attention,**

**A comma appears between data.**

**The files are sorted by serial code in ascending order.**

For example, the series"Money Heist"

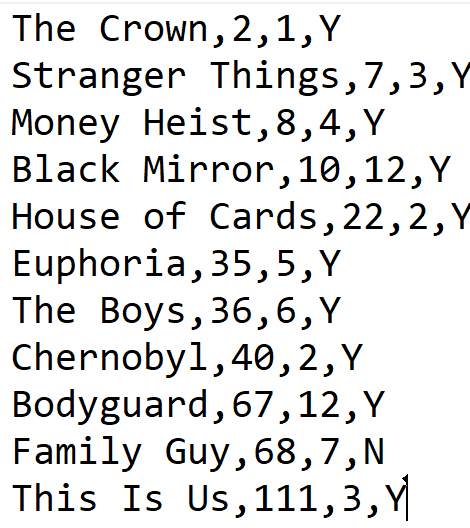
The serial code is8

The number of seasons is:4

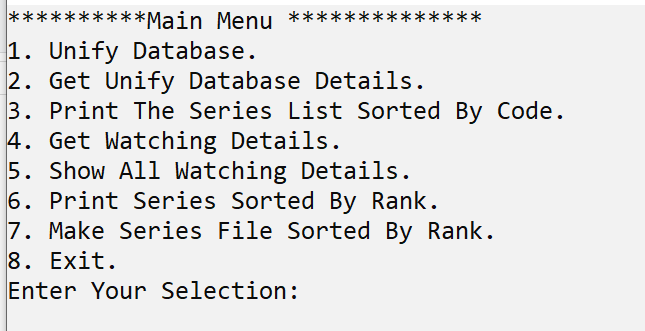
The series is active

The line corresponding to this series in the file will look like this:

Money Heist,8,4,Y

Below is an example of the series file:

Upon starting the program, the following menu will be displayed:



**Option 1 in the menu:Unify Database**

This option must be done at the beginning.

Its purpose is to unite the data of the 2 files, sorted by serial code, into one file. The merging of the 2 fileswill be carried out as follows:

If the series appears in only one of the files,

If the series is not active => no line will be written to the file

If the series is active => a line will be written to the file with the series data

If the series appears in the 2 files, each of the following options must be handled:

1. twoThe series is inactive =>No line will be written to the file
2. Both series are active => a line will be written to the file so that the number eseasonswill be the higher of the two.
3. Only one series is active => a line will be written to the file with the data of the active series.

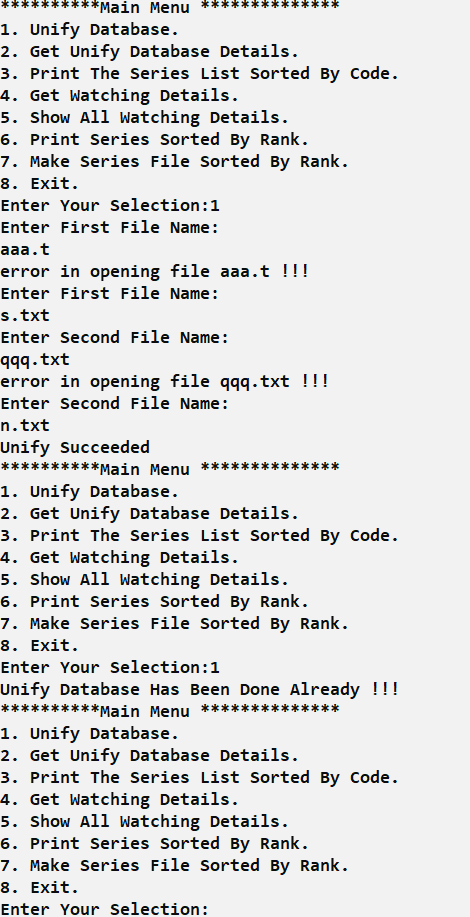
The name of the output file will be: **series.txt**

With the selection of this option, the user must enter the file names of each of the companies.

If there is a problem opening the files, an appropriate message will be displayed and the menu will be displayed again.

Below is an example of using option number 1:

The file names are,n.txt and s.txt.



**Option 2 in the menu:Get Unify Database Details**

Upon selecting this option in the menu and provided that option 1 was performed, the program will read the file dataseries.txt

The data will be stored in memory into an array of pointers.

In addition, the file**watching.txt**into which the viewing data will be written will be initialized again as an empty file.

The size of the array is defined dynamically according to the number of series, each cell in the array is a pointer to a structure of the following form:

typedef struct serie{

charseriesName[20];

intnumSeasons;

intrank

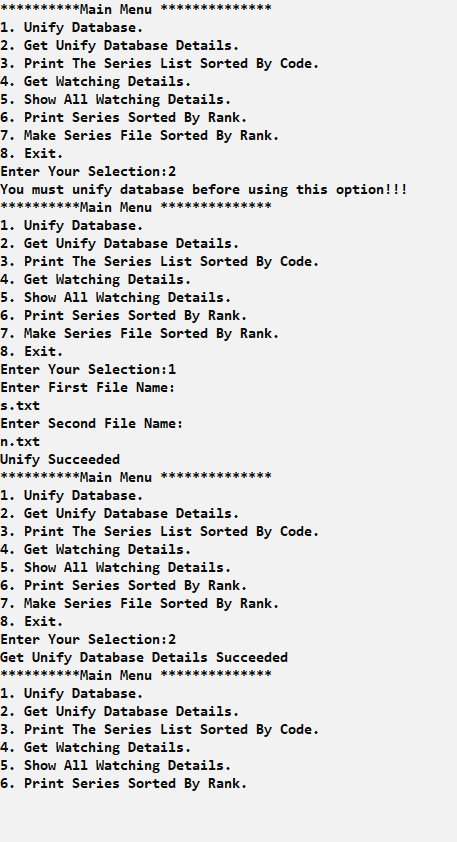
int\* watchingDetails;

}serie;

For each series, the name of the series, its code, the number of seasons and also an array sized as the number of seasons will be saved.

At the end of the operation, an appropriate message will be displayed and the menu will be displayed again.

Below is an example of using the number option2:



**Option 3 in the menu:**Print The Series List Sorted By Code

With the selection of this option in the menu and provided that option 1 and 2 have been carried out, the program will display the series data sorted by series code in ascending order.

Below is an example of using the number option3:



**option4 in the menu: Get Watching Details**

Upon selecting this option in the menu and provided that option 1 and 2 have been carried out, the program will receive viewing data and update the memory database and the viewing file named: **watching.txt**

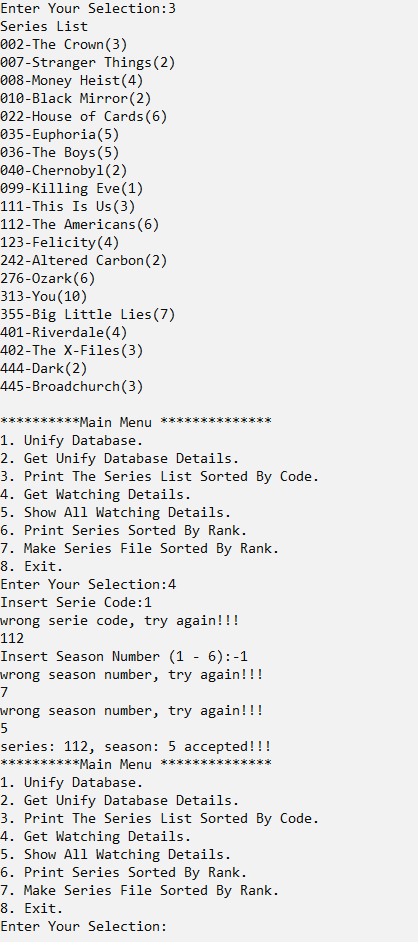
When receiving, first the series code will be received and then the season number,

If an incorrect code is entered, the message will be displayed:wrong series code, try again!!!

If an incorrect season number is entered, the message will be displayed:wrong season number, try again!!!

If correct input is received, the message will be displayed:series: %d, season: %d accepted!!!

Below is an example of using the number option4:

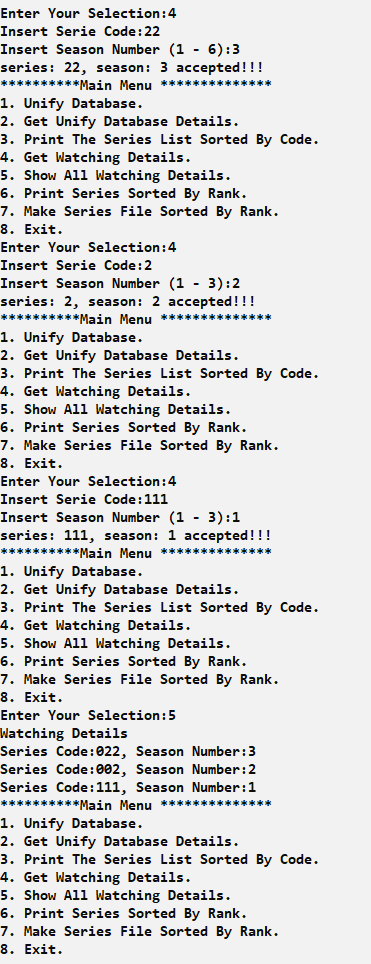


**option5 in the menu: Show All Watching Details**

With the selection of this option in the menu and provided that option 1 and 2 have been carried out, the program will display the recorded viewing data.

For each input data, an output line will be displayed that includes the series code and the season number.

Below is an example of using the number option5:

****

**Option 6 in the menu:Print Series Sorted By Rank**

With the selection of this option in the menu and provided that option 1 and 2 have been carried out, the program will display the data of the series sorted according to their ranking in descending order. That means the series with the highest rating will be shown first and so on until the lowest rating. The rating of the series is the total number of views in the series (in all seasons).

For example:

for the seriesThis Is Us, which has 3 seasons, and according to the viewing data recorded,

In season 1: there were 5 views, in season 2: there were 3 views and in season 3: there were 2 views, for a total of 10 views.

Therefore the rating of the series is 10.

First the rating will be displayed, and then all the series in this rating will be displayed (sorted by series code in ascending order).

For each series, a title line containing the series code and its name will be displayed, and then for each season in the series, the season number and the number of views that were in this season will be displayed.

Below is an example of using the number option6:



**Option 7 in the menu:Make Series File Sorted By Rank**

Upon selecting this option in the menu and provided that option 1 and 2 have been carried out, the program will write into a file named

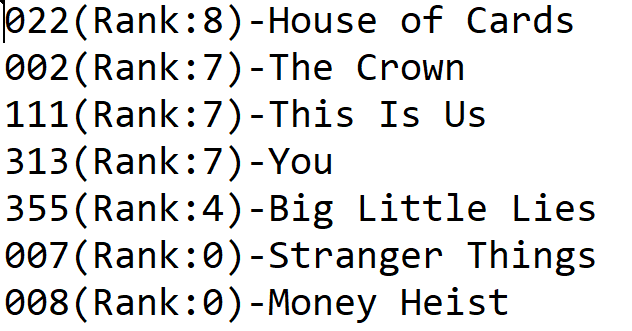
seriesRank.txt the data of the series sorted according to their ranking in descending order.

The data will be written to the file as follows:

For each series, a title line containing the series code and its name and rating will be displayed,

After that, for each season in the series, the season number and the number of views that were in this season will be displayed.

Below is an example of a file created after selecting this option:



**Option 8 in the menu:Exit**

With this option selected, the program will release the captured memory areas,will display the following message:

End Of The Program...and it will end.

Here are the definitions of the structures and a prototype of the functions, the function signatures can be changed:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct serie{

charseriesName[20];

intnumSeasons;

intrank

int\* watchingDetails;

}serie;

intshowMenu();

intunifyDatabase(FILE\* ,FILE\* );

serie\*\* getUnifyDatabaseDetails();

voidprintSeriesList(serie\*\* );

voidgetWatchingDetails(serie\*\*,int\* );

voidprintWatchingDetails();

voidprintSeriesByCode(serie\*\* );

voidprintSeriesByRank(serie\*\*,int);

voidmakeSeriesFileSortedByRank(serie\*\*,int);

voidmain() {…

**Discounts and highlights:**

* It can be assumed that the serial files exist and are normal. The file name is a maximum of 19 characters.
* Each of the above files must be read only once.
* Care must be taken for proper memory management and the release of the memory at the end of the work. You must use dynamic memory allocation.
* In any case of inconsistency between the written instructions and the behavior of the runtime file–Act according to the run-in file and inform the responsible practitioner of the discrepancy.
* Sample input and output files are attached. In some cases there is no line break in the output printing. A new line must be started on each printout–according to the behavior of the runtime file.

**Successfully!**