

## *CPS1011 Assignment Documentation*

Github repo:

<https://github.com/aidansey164/CPS1011AssignmentDirectory.git>

### 1. Problem Solving

A i) To implement function `init_array(array ar, size)`, the user is instructed to enter the values using `scanf`, that will be contained inside the array, that are dependent on the size of the array inputted in the parameter variable `size`. Then the elements that the user inputted in the array are outputted to the screen. Finally the size of the array is returned also as the last string.

```
Please enter 5 values:  
1  
2  
3  
4  
5
```

```
Elements in array are: 1 , 2 , 3 , 4 , 5 , Size of array is 5
```

The user enters the 5 values as seen in the above first image of the output, then the elements are printed to the user's screen in order whilst at the end, the size of the array is also printed to the screen.

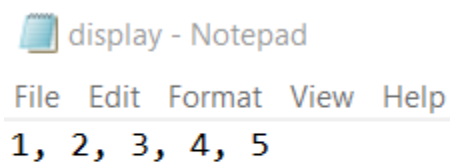
ii) To implement function `display(array ar, size)`, firstly the user has a choice of either outputting to the screen or outputting the array to a file using a switch statement in the function. If case 1 is chosen, then the array is outputted to screen in the format of `{offset, value}`. Whilst if case is 2, firstly a new txt file is created and the array is written to this txt file.

```
Press 1 to print to screen or 2 to print to a txt file
```

The user has the choice of either printing to screen or to an external text file.

```
offset: '04'  
value: '5'
```

Part of the output if the user chooses to print to the screen.



A screenshot of a Notepad window titled "display - Notepad". The window has a menu bar with "File", "Edit", "Format", "View", and "Help". The text content of the window is "1, 2, 3, 4, 5".

The output if the user chooses to print to a file.

iii) To implement function `reverse(array a, array b, size)`, a for loop loops through array a, and reverses the structure of the elements to array b. It's important that arrays b and a are of the same size. If array b already has elements inside it, these elements will be removed and replaced with the reversed elements of array a.

```
Array B already has elements inside it 1 1 4 4 5
```

This output shows that array B already has elements inside of it, these elements will be discarded and be replaced with the reversed order of array A.

Original	-->	Copy
5	-->	1
4	-->	2
3	-->	3
2	-->	4
1	-->	5

This output shows the contents of original array (A) and copy array (B). Copy array contains the reversed order elements of the original array A. If the arrays are not of the same size, the function will not work properly.

```
Array B already has elements inside it 1 5 4 3 2
Original      --> Copy
2            --> 1
3            --> 2
3            --> 3
2            --> 3
1            --> 2
```

In this scenario array B contains only 1 element which is 1, resulting in the function to malfunction and output elements that do make sense, but the original array was effected and thus completely changed the elements inside of it. Array B resulted in copying the elements of A, 5 4 3 2 to adequate for the different sizes and get both arrays to have the same sizes to calculate the reverse order into array B.

iv) To implement function frequency(), a struct freqarr was created that holds two arrays, one that holds the inputted array arr() in the parameter and the other array that holds frequency values freq(). The size of these arrays depends on the size of the array specified in the function parameter. A for loop loops throughout the array and checks for duplicate elements in the array. A counter count was set to be incremented whenever there is more than one frequency of that number. Finally, the two arrays were merged into one single array, first by merging the arrays into an array and then removing the unnecessary 0 values that were created when merging.

```
1 1 1 25
```

The input array, where the expected output is 3 frequencies of 1 and 1 frequency of 25.

```
The Frequency of the elements in this Array is :  
{1 , 3} {25 , 1}
```

The first element is the element value and the second is the number of frequencies of that value.

```
Merging two arrays in structure into single array  
1 3 25 1
```

The merged array of the two arrays is also outputted with no 0 values present in the array.

v) The implementation of function `display_frequency()` was quite similar to `display()`. The user has a choice of either printing to the console or to an external text file.

```
Press 1 to print to screen or 2 to print to a txt file  
1
```

```
{ {Number: 3 }  
  {Frequency: 2} }  
  
{ {Number: 2 }  
  {Frequency: 1} }
```

The output if the user selects choice 1.

```
displayfrequency - Notepad  
File Edit Format View Help  
3 2 2 1
```

The output if the user selects choice 2

B) To implement the command line, a switch statement was implemented where beforehand the user is presented with a menu with each number corresponding to running a function from above. The default statement at the end of the switch assures that the user can try again if the user fails to enter the correct number corresponding to a case in the switch statement.

```
1. init array()
2. display()
3. reverse()
4. frequency()
5. display_frequency()
6. Exit
Enter your choice :5
```

The menu presented to the user.

```
Enter your choice :7
Invalid input, Try again
```

Output if the user enters a number out of range. The menu is then presented again for the user to enter the correct number.

## 2. A Datatable library

A i) To implement initDT(), firstly a struct called Datatable was created which contains 4 variables. The first variable scold holds all string values. The second variable floatcols holds all float values in the columns. The third variable colnames holds the label names for the columns. Finally, the rowcounter variables hold the number of rows in the Datatable structure. The structure is allocated memory, if the memory allocation is unsuccessful then the user will get

an EXIT\_FAILURE. If memory allocation is successful, the user is instructed to enter the column name labels. Then the user has to enter the number of rows, and finally the user has to enter the data into the datatable, with the row counter incrementing accordingly. In the end the pointer to the datatable structure is returned.

```
Allocation of memory successful
Please enter name label for Column 1
```

After memory is allocated, the user is instructed to enter the 6 column label names and number of rows together with the data inside the rows.

```
Enter No of Rows
2
```

```
Enter data for Column 1 and Row 2
2.25
```

ii) To implement deinitDT(), the free() method was used to free the allocated memory of the datatable structure.

iii) The function loadDT() was not implemented.

iv) To implement exportDT(), first the user file inputted in the parameter is loaded using the fopen() method. The user can either create a new file by typing a name for the file ending in the format of .csv or load in a file that is already in the same directory. Necessary checks were implemented to check that the file was opened correctly or not. The column label names were firstly printed using fprintf(). Then subsequently the rest of the rows were printed to the csv file. This is how the datatable looks like when exported to the file. The columns are separated by a comma accordingly.

Column1	, Column2	, Column3	, Colum4	, <u>Column5</u>	, <u>Column6</u>
1.230000	,1.250000	,1.260000	, Test1	, Test2	, Test3
2.120000	,2.130000	,3.120000	, Test4	, Test5	, Test6

v) To implement showDT(), 6 arrays were initialized and the column label names were copied inside the array in order to print the column label names. The while statement indicates till which row will be printed, which is set to the 15th row. Then the data is printed by incrementing the columns in order to print every single column. An integer i is incremented to print till the last row. The other printf() statement is used to print the other rows that were left empty by the user and thus do not contain any useful information.

Column1	, Column2	, Column3	, Colum4	, Column5	, Column6
1.230000	,1.250000	,1.260000	, Test1	, Test2	, Test3
2.120000	,2.130000	,3.120000	, Test4	, Test5	, Test6
0.000000	,0.000000	,0.000000	,	,	,
0.000000	,0.000000	,0.000000	,	,	,

This is how the output looks like, the empty rows keep going till 15 rows are printed.

vi) To implement projectDT(), firstly the user has to enter 4 integers, firstly the start row and end row(x, y). Then the start column and end column (m, n). According to these values the datatable is printed respecting these restrictions, by first a while loop prints the column label names and another while loop prints the rows as inputted by the user. They are copied to the new datatable structure, with its new row counter incremented accordingly. The output of the copied datatable structure was not correct.

vii) To implement mutateDT(), the user has to enter an integer in the parameter which will be used to mutate the float values. Using a switch statement, the user has the choice of manipulating the values or text. The user can either multiply the float numbers or divide the float numbers by the number inputted by the user. On

the other hand, the user can convert all strings in the columns to uppercase format.

```
-----Values Manipulation-----
1. Multiply Float Numbers by number inputted in parameter
2. Divide Float Numbers by number inputted in parameter
-----String Manipulation-----
3. Convert Strings to Uppercase
4. Exit
1
```

The menu presented to the user, with the user choosing option 1.

Column1	, Column2	, Column3	, Colum4	, Column5	, Column6
12.300000	,12.500000	,12.600000	, Test1	, Test2	, Test3
21.200000	,21.300000	,31.200000	, Test4	, Test5	, Test6

The output data after mutating the float values multiplied by 10

Column1	, Column2	, Column3	, Colum4	, Column5	, Column6
12.300000	,12.500000	,12.600000	, TEST1	, TEST2	, TEST3
21.200000	,21.300000	,31.200000	, TEST4	, TEST5	, TEST6

The output data after mutating the text to uppercase.

C) To compute the whole implementation as a shared library, `add_library()` and `target_link_libraries()` were added to the `cmakelist` in order to create the `.dll` files to dynamically link the library.

```
add_library(Datatable SHARED Datatable.c)
target_link_libraries(CPS1011_Task2 Datatable)
```