**OS\_PRE\_PROGRAMMING\_LAB\_ASSIGNMENT**

**Submission by:**

**T Karthikeyan**

**CED18I064**

All my codes codes can be seen pressing this below link

[**PRESS ME**](https://drive.google.com/drive/folders/1W1y1OlzD7YG24M6L2GGZmUVRYlKeA5F2?usp=sharing)

**I) Develop an application (using C & Command Line Arguments) for:**

**i) Simulate the behavior of cp command in linux. (you should not invoke cp**

**command from your C source!). Also your application should validate right**

**usage; if less or more number of arguments are passed to the executable the**

**program should prompt a message to the user. File read and write function**

**calls are allowed. Rename your executable as mycopy.**

**Example usage cud be ./mcyopy fact.c factcopy.c**

**LOGIC**

The C program given below is a simulation of the Linux “cp” copy command. The copy command can be used to make a copy of your files and directories, but here in this implementation only the basic functionality of copy the content from one file to another is handled.

The program for “cp” command takes in two arguments namely the source file and the destination file.

./cpcmd sourcefile destinationfile

1) First check if both source & the destination files are received from the command line argument and exit if argc counter is not equal to 3.

There is also a check to handle “--help” option to print the usage of cpcmd.

2) Open the source file with read only flag set.

3) Open the destination file with the respective flags & modes.

O\_WRONLY -> Open the file in write only mode

O\_TRUNC -> Truncates the contents of the existing file

O\_CREAT -> Creates a new file if it doesn’t exist

S\_IXUSR are file permissions for the current user (‘X’ can be R for read & W for write)

S\_IXGRP are file permissions for the groups (‘X’ can be R for read & W for write)

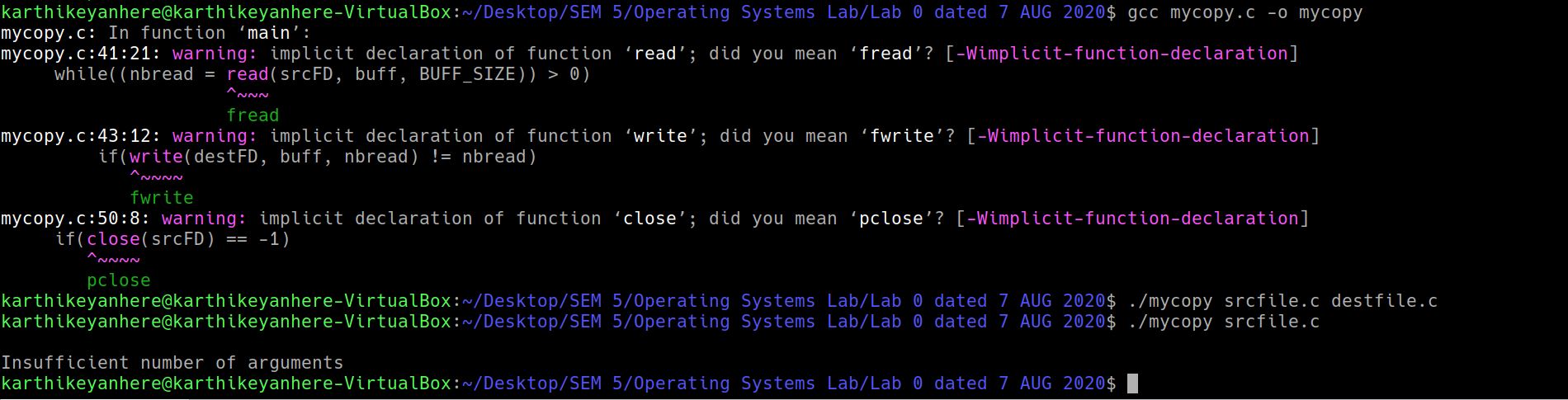
S\_IXOTH are file permissions for the groups (‘X’ can be R for read & W for write)

4) Start data transfer from source file to destination file till it reaches EOF (nbread == 0)

**COMMANDS USED**

terminal~ g++ mycopy.c -o mycopy

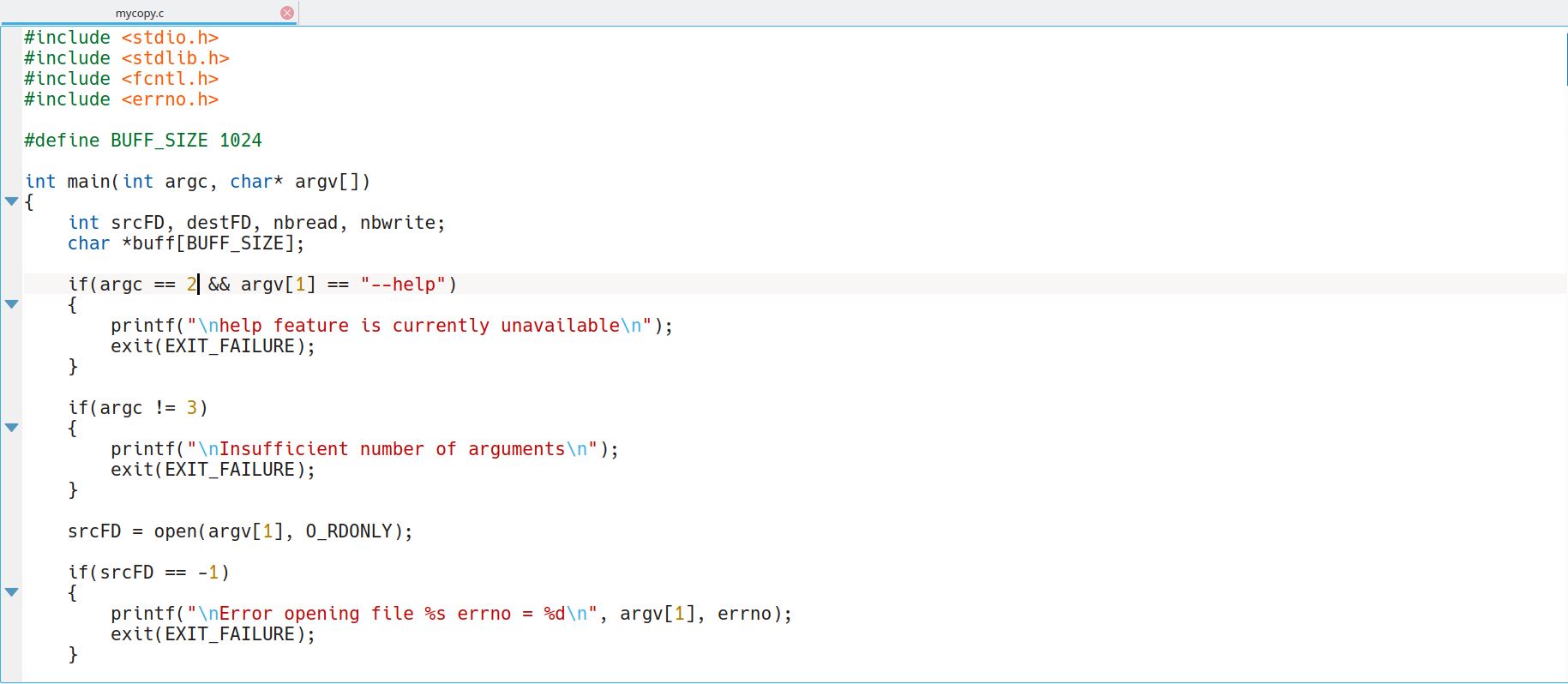
terminal~ ./mycopy srcfile.c destfile.c

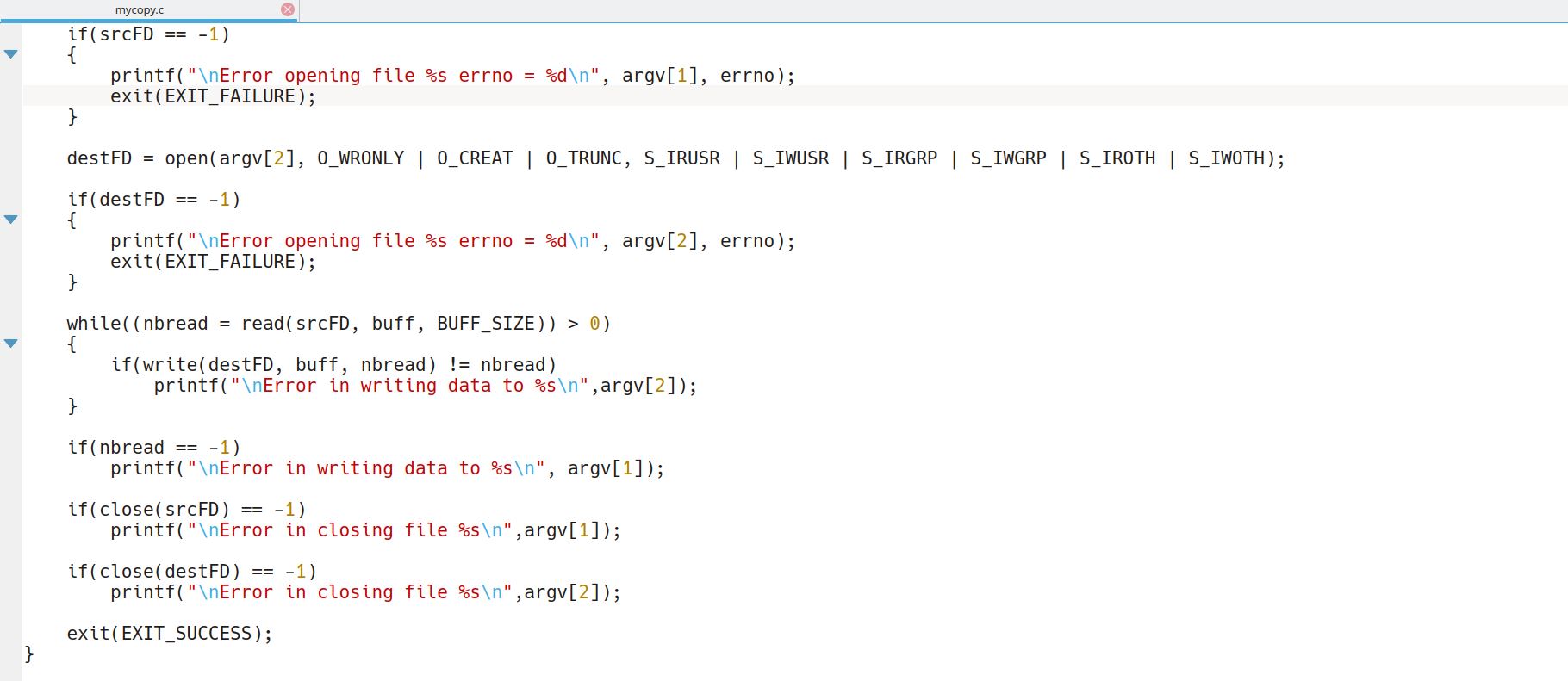


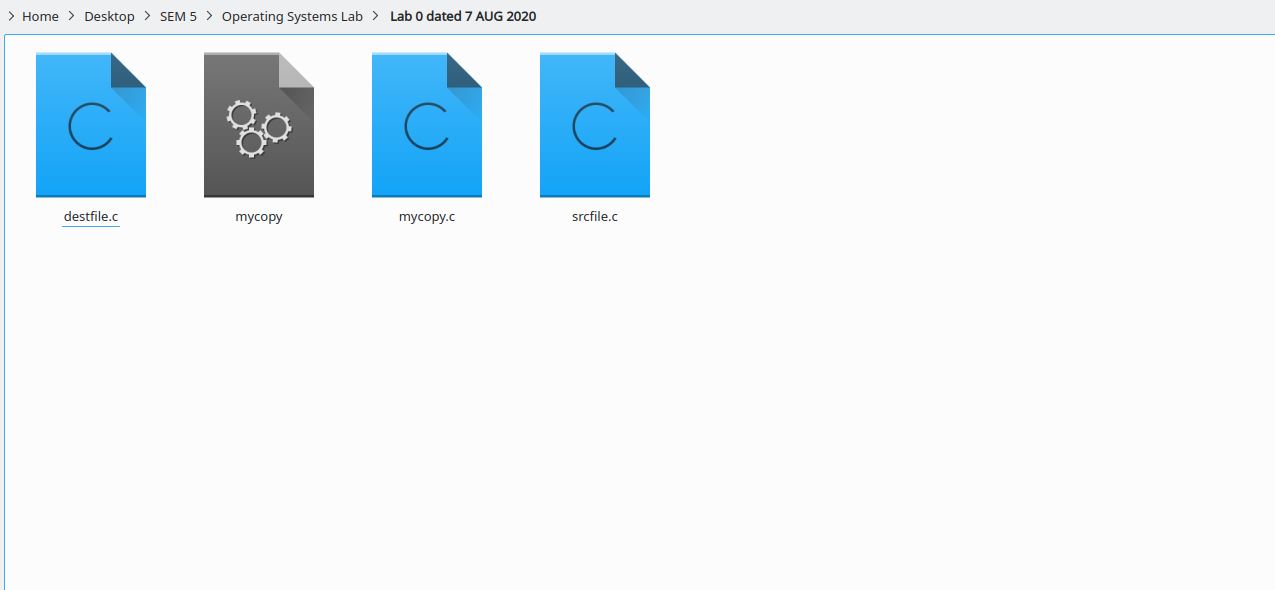
terminal~ ./mycopy srcfile.c

Insufficient number of arguments

**CODE**

****

****

****

**ii) Extra Credits Qn – Extend the above application / develop an application to**

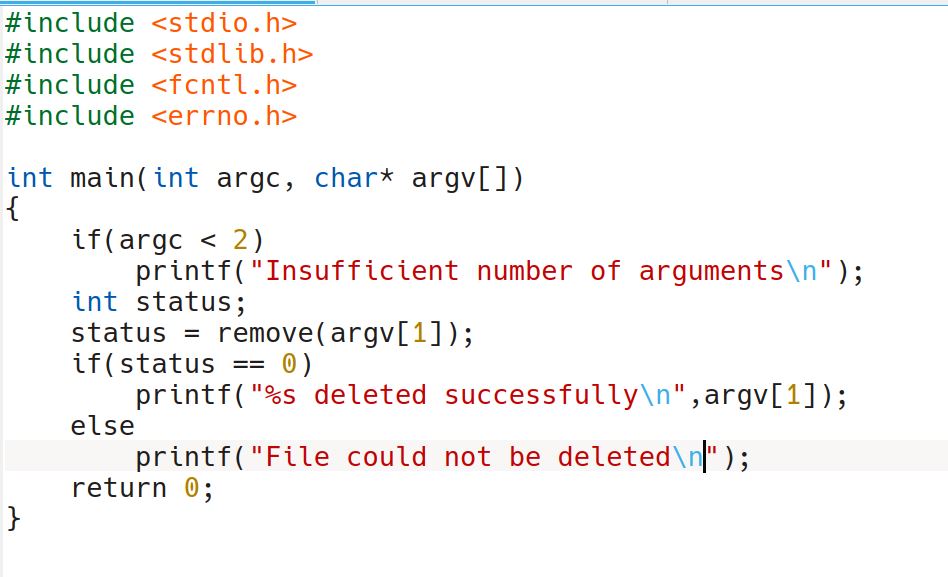
**simulate the behavior of rm command in linux. rm command invoke from**

**Source is not allowed! Other features as in earlier application to be supported.**

**LOGIC**

Using argc and argv in the main function, making use of C remove function, passing the argv[1] as a parameter into remove function.

**CODE**

****

**COMMANDS**

terminal~./myremove destfile.c

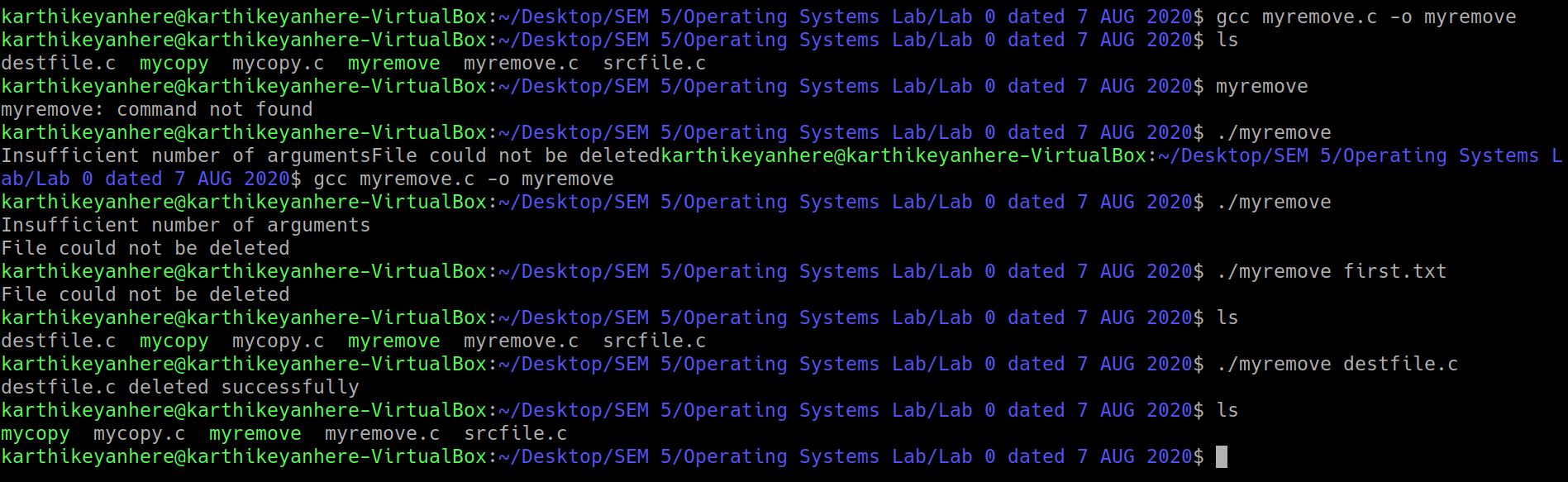
destfile.c deleted successfully

Terminal~ ./myremove

Insufficient number of arguments

File could not be deleted

**OUTPUT**

****

**II ) Develop an application (using C & Command Line Arguments) for:**

**i) Sort an array of varying number of integers in ascending or descending order.**

**The array and array size are passed at command line. Invoke of linux command**

**sort is not allowed. Use of atoi or itoa fns is allowed (need you should read online**

**resources!). Let your program handle invalid usages as well!**

**Eg. ./mysort 5 1 50 40 30 20 1 here 5 is array size and 1 means ascending order**

**sort and the rest of the input is the array to be sorted. Your code should handle**

**descending order sort as well.**

**Extra Credits Qn: (I wud advise everbody to try!)**

**Can you implement the above sorting (both ascending or descending) using only**

**function internally for sorting logic ( I mean bubble or insertion etc..)You should**

**define the logic in your source code only once but the application should be able**

**to handle both ascending or descending order sort!). Hint use function pointers!**

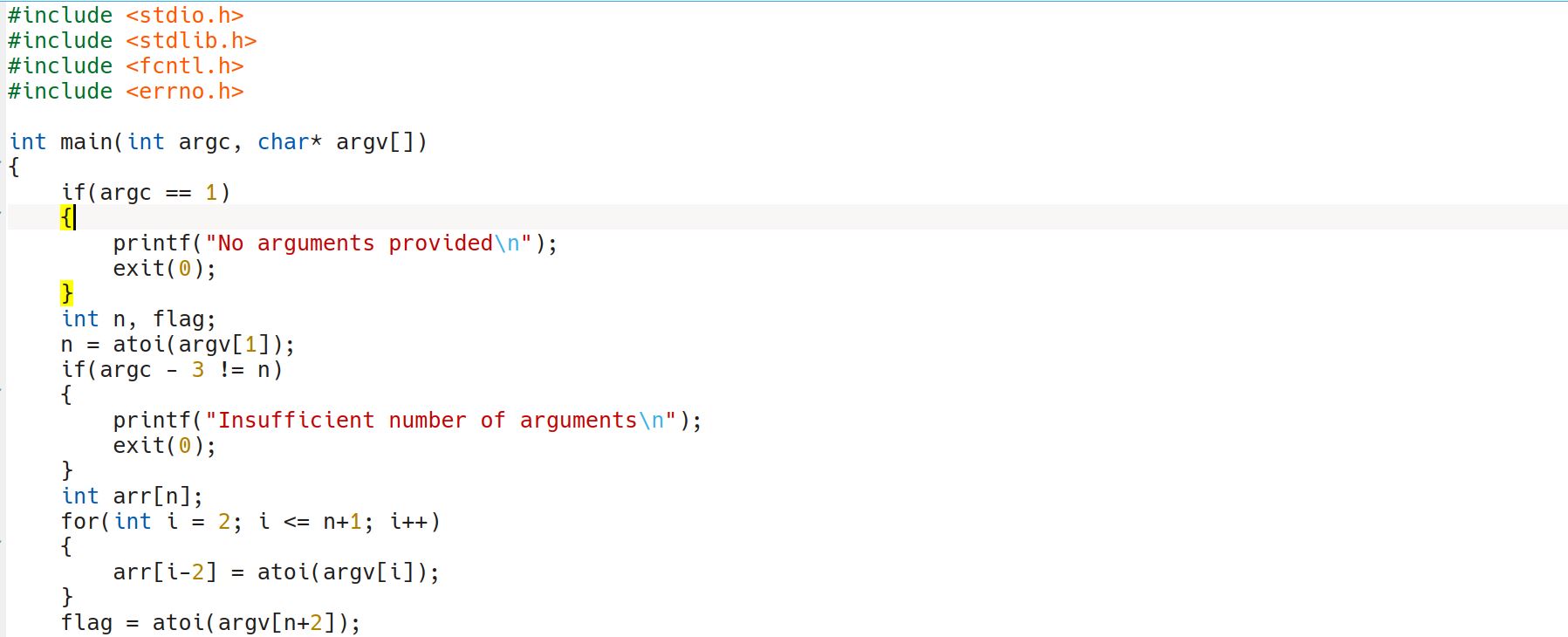
**LOGIC**

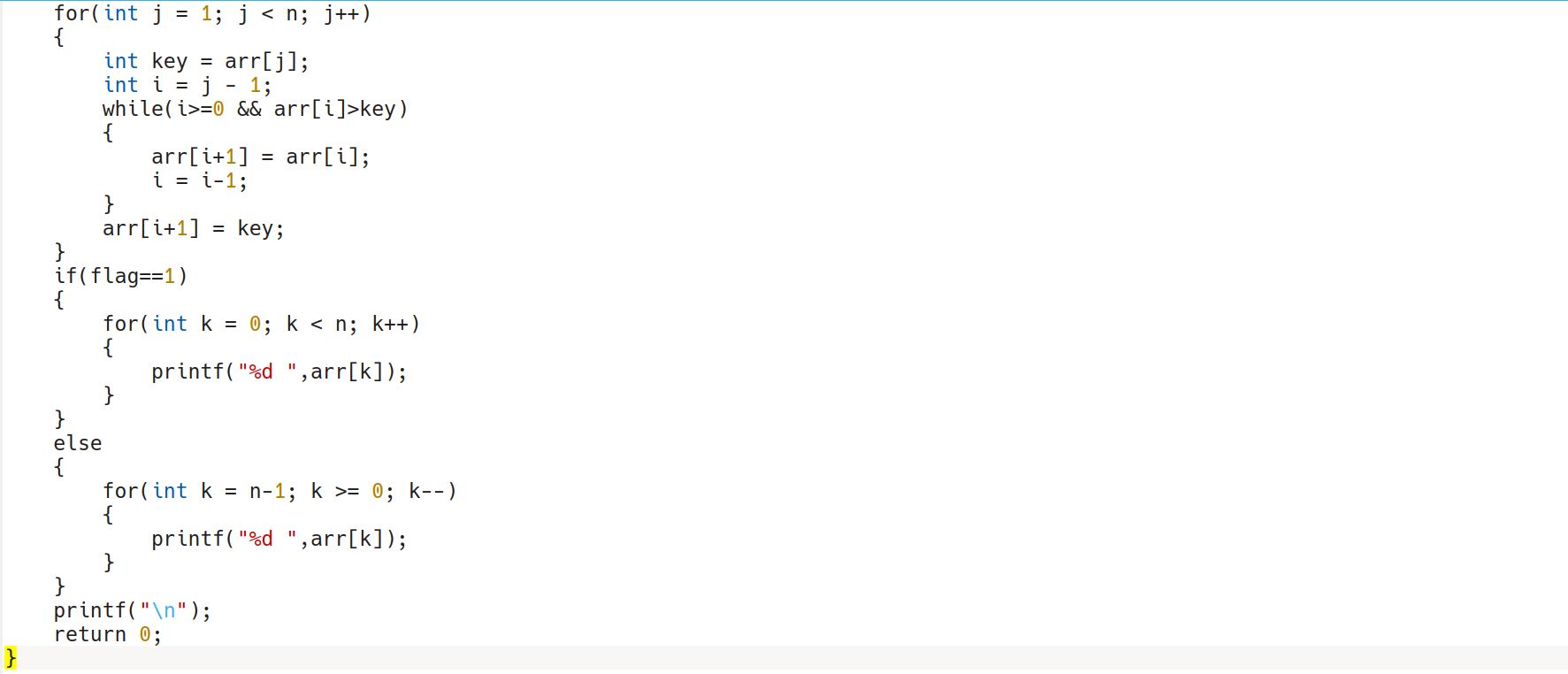
Using command line arguments, number of elements in an array, elements of array along with a flag which specifies whether the array should be sorted in ascending or descending order.

Using atoi function, string is converted into integer and then sorting is performed.

Insertion sort subroutine is used.

**CODE**

****

****

**COMMANDS**

terminal~ ./mysort 5 5 3 4 2 1 1

1 2 3 4 5

terminal~ ./mysort 5 5 3 4 2 1 2

5 4 3 2 1

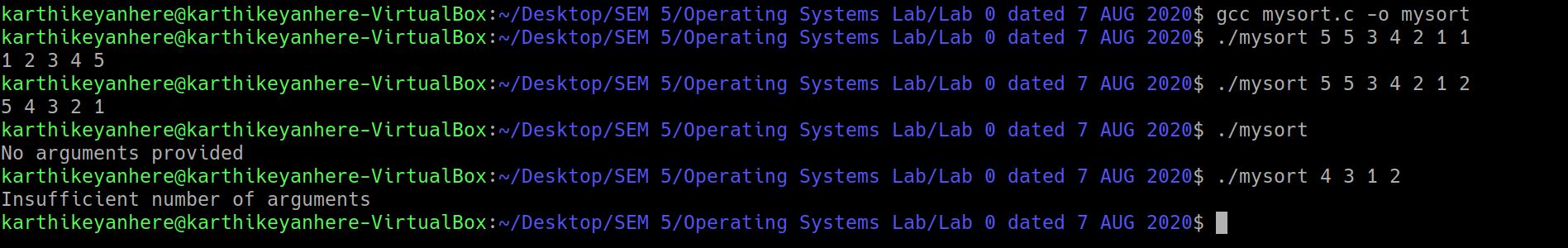
terminal~ ./mysort 4 3 1 2

Insufficient number of arguments

terminal~ ./mysort

No arguments provided

**OUTPUT**

****

**III ) Develop an application (using function overloading & command line**

**arguments in C) for:**

**a) Sorting an array of integers or floating point or characters passed at command**

**line. Usage syntax you can follow a similar style as for the II question and also**

**support validation logic in the code.**

**LOGIC**

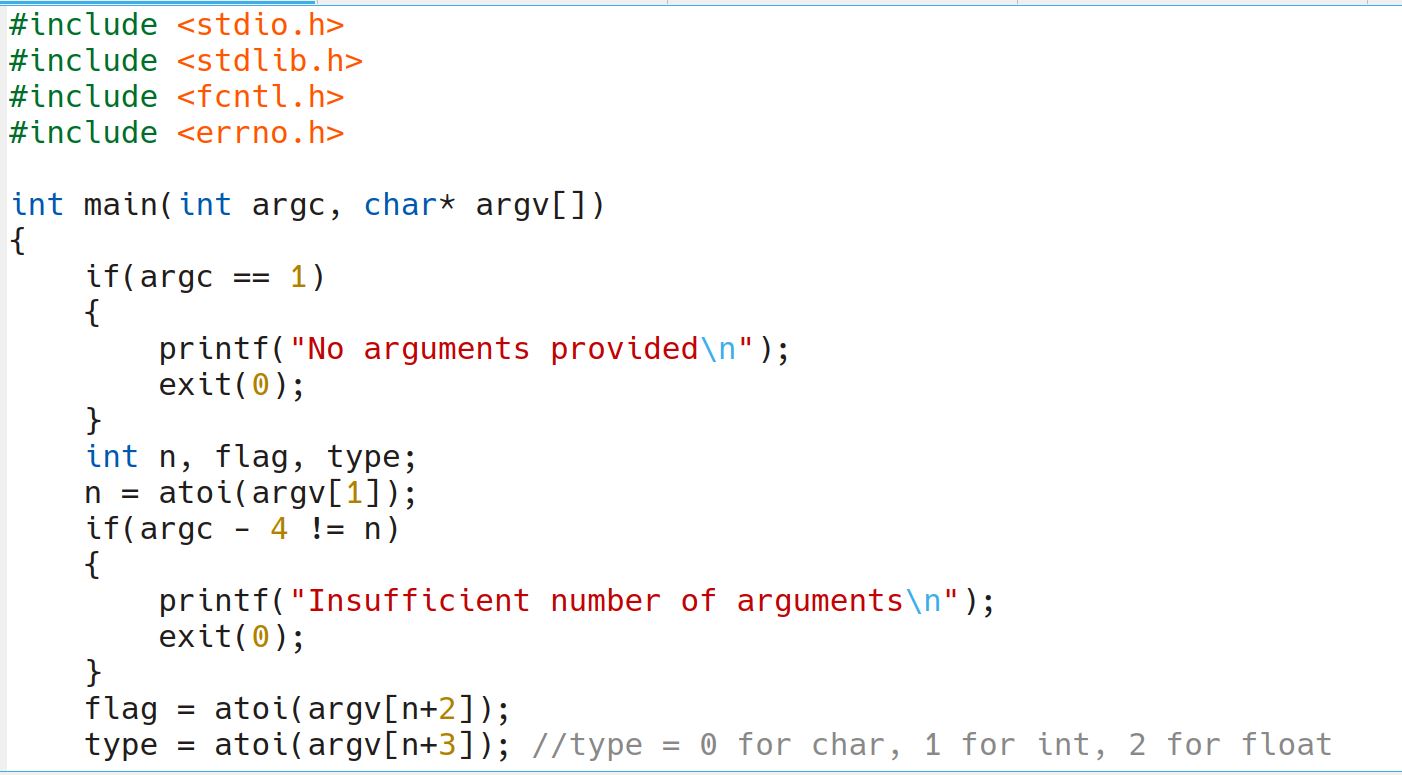
function overloading is implemented with the help of one extra parameter

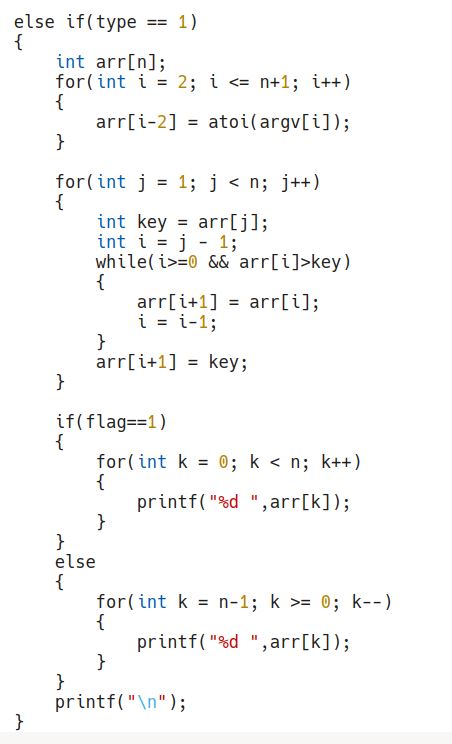
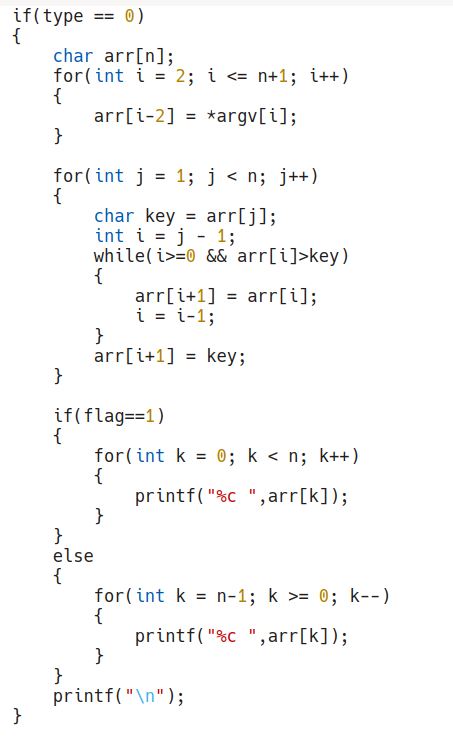
ex ./mysortfo 5 b x r u w 1 0

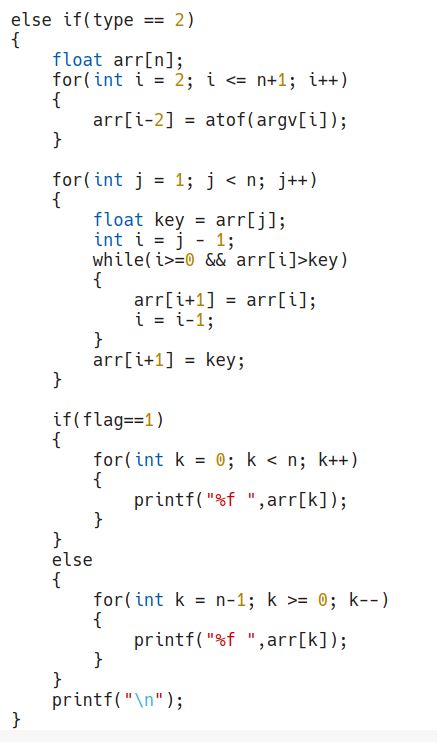
In the above example, the last argument specifies that the elements of the array are characters, the second last argument (1 for ascending , else descending) specifies that sorting should be ascending.

|  |  |
| --- | --- |
| 0 | character |
| 1 | integer |
| 2 | float |

**CODE**

****

****

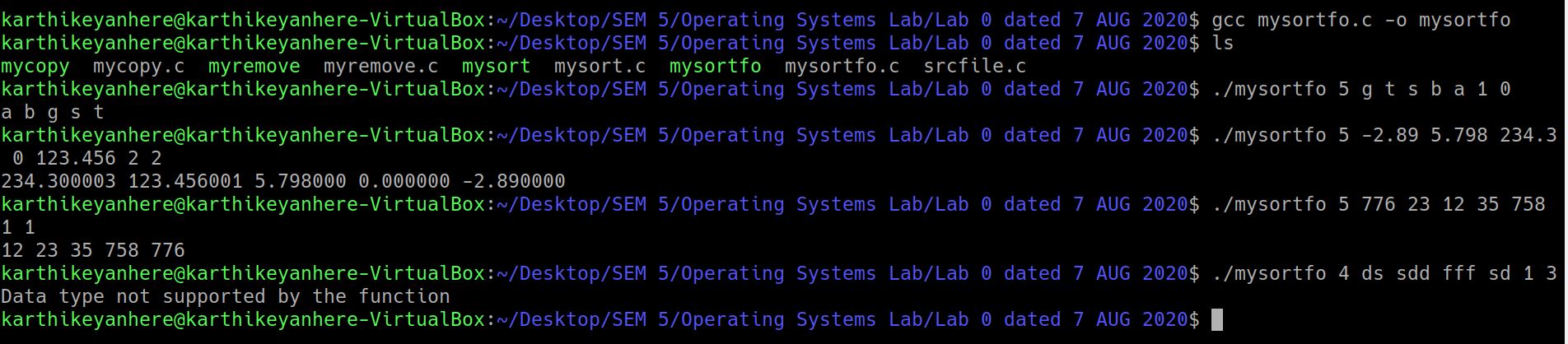
****

**COMMANDS**

terminal~ ./mysortfo 5 g t s b a 1 0

a b g s t

**OUTPUT**

****

**IV ) Develop an application (using function templates & command line**

**arguments in C) for:**

**Same as above but you should define sort function only once internally and leave**

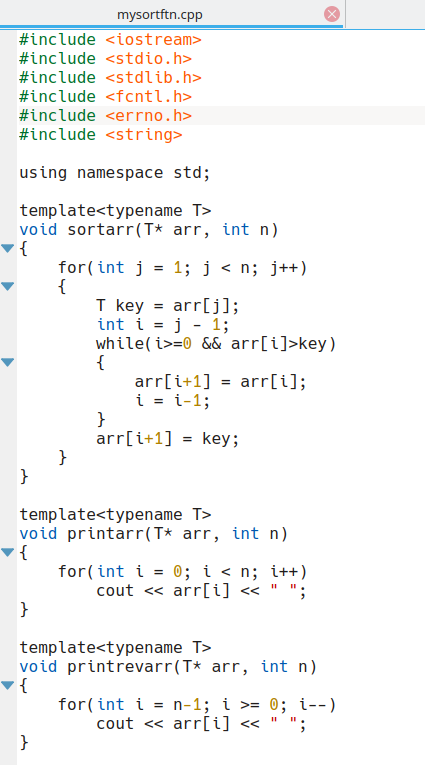
**it to the compiler to generate data type specific functions. Clue is to use function**

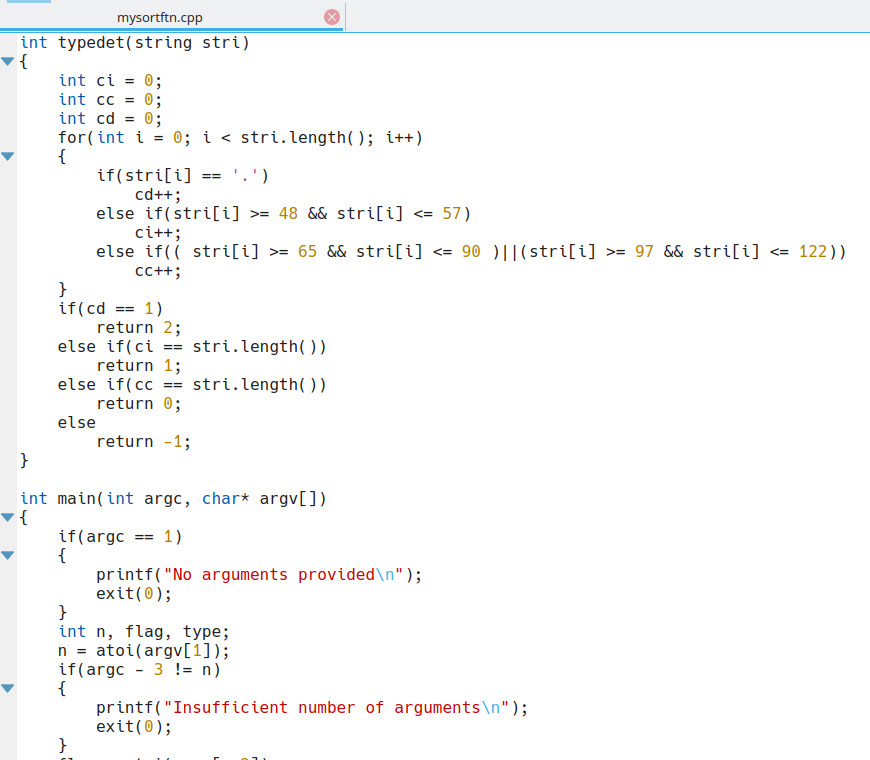
**templates feature in C. Read on it more!**

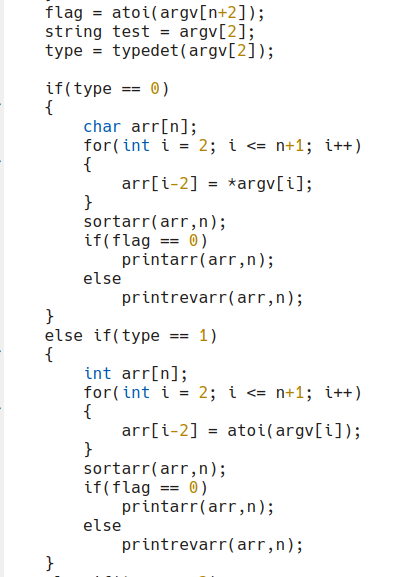
**LOGIC**

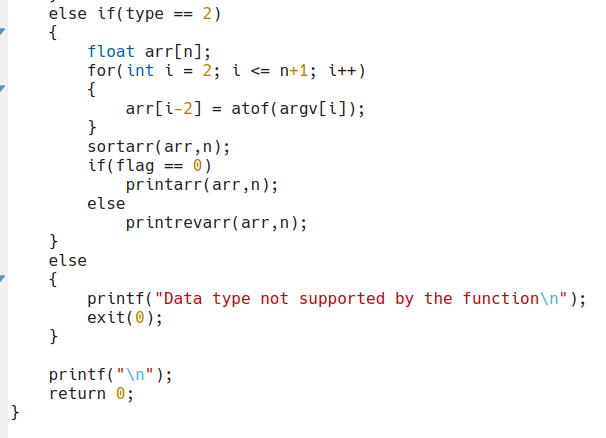
Function templates concept is implemented in C. A argv (string) is identified as a character array or integer or floating point number based on ascii values of argv(string).

**CODE**

****

****

****

****

**COMMANDS**

terminal~ ./mysortft 5 1 5 2 3 4 1

1 2 3 4 5

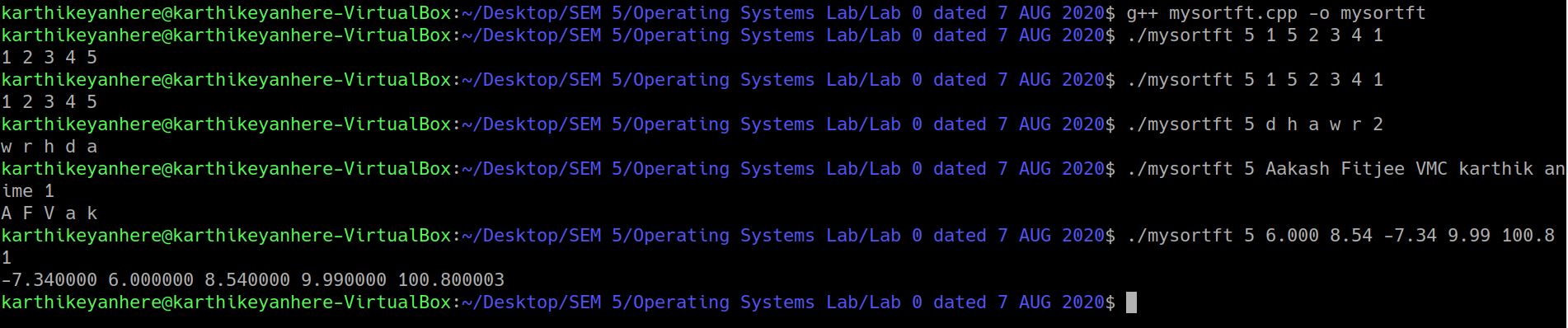
terminal~ ./mysortft 5 d h a w r 2

w r h d a

terminal~ ./mysortft 5 6.000 8.54 -7.34 9.99 100.8 1

-7.34 6.000 8.54 9.99 100.8

**OUTPUT**

****