OS_PRE_PROGRAMMING_LAB_ASSIGNMENT

Submission by:
T Karthikeyan
CED18I064

All my codes codes can be seen pressing this below link
PRESS ME

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)

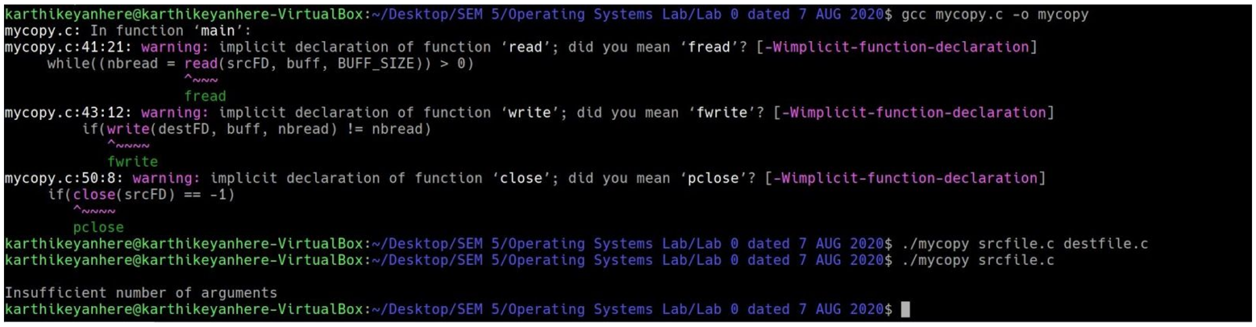
S_IXOTH are file permissions for the groups (‘X’ can be R for read & W for write)

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
```

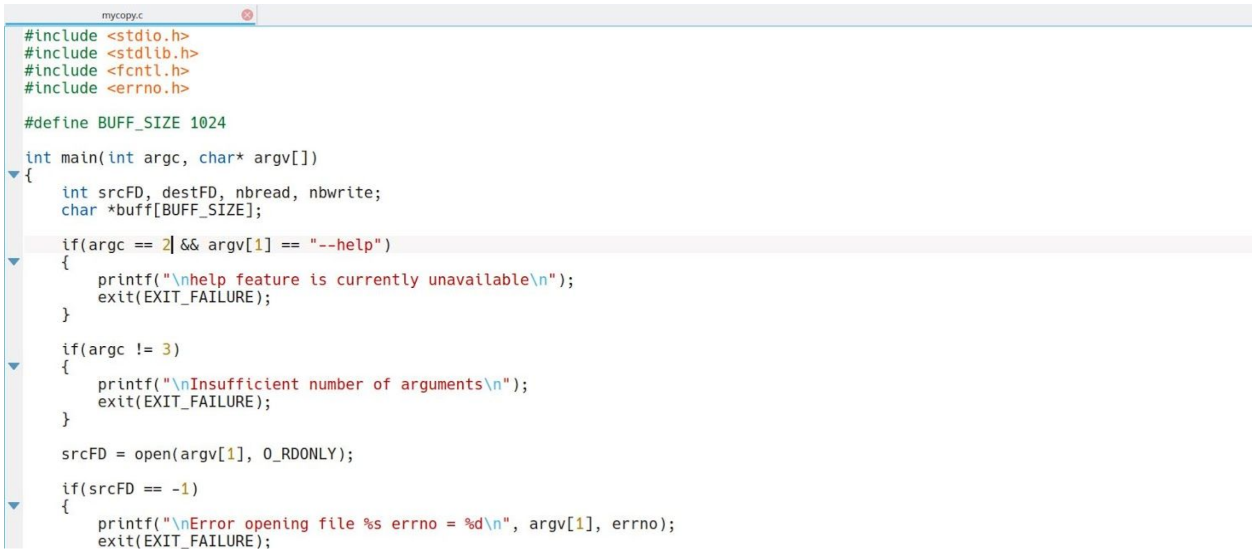


```
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ gcc mycopy.c -o mycopy
mycopy.c: In function ‘main’:
mycopy.c:41:21: warning: implicit declaration of function ‘read’; did you mean ‘fread’? [-Wimplicit-function-declaration]
     while((nbread = read(srcFD, buff, BUFF_SIZE)) > 0)
                     ^~~~
                     fread
mycopy.c:43:12: warning: implicit declaration of function ‘write’; did you mean ‘fwrite’? [-Wimplicit-function-declaration]
         if(write(destFD, buff, nbread) != nbread)
            ^~~~~
            fwrite
mycopy.c:50:8: warning: implicit declaration of function ‘close’; did you mean ‘pclose’? [-Wimplicit-function-declaration]
     if(close(srcFD) == -1)
        ^~~~~
        pclose
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ./mycopy srcfile.c destfile.c
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ./mycopy srcfile.c

Insufficient number of arguments
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$
```

terminal~ ./mycopy srcfile.c

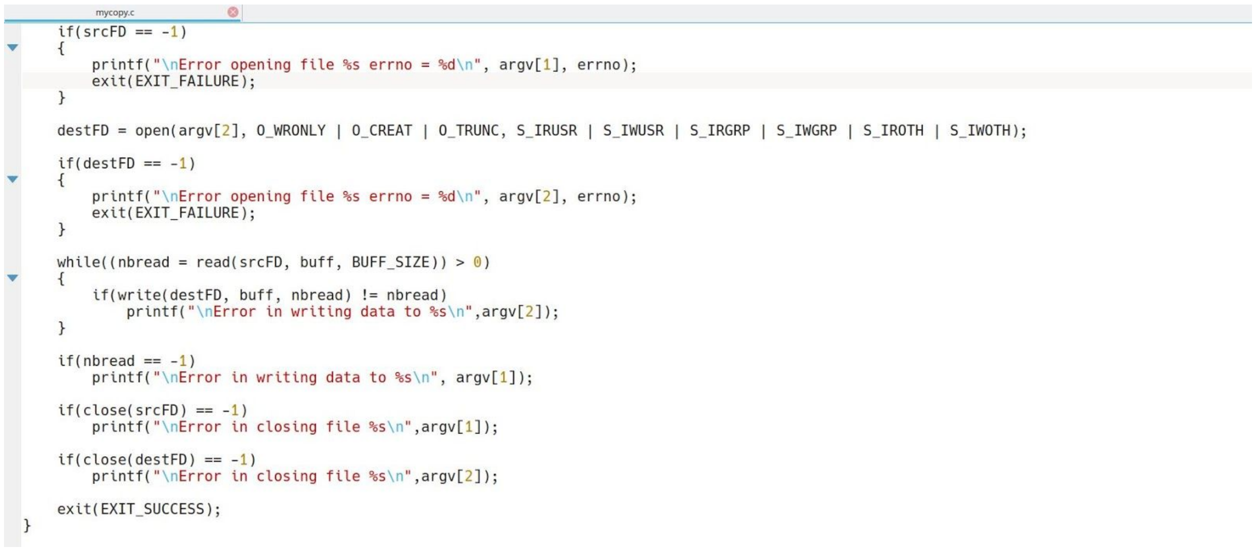
Insufficient number of arguments

CODE

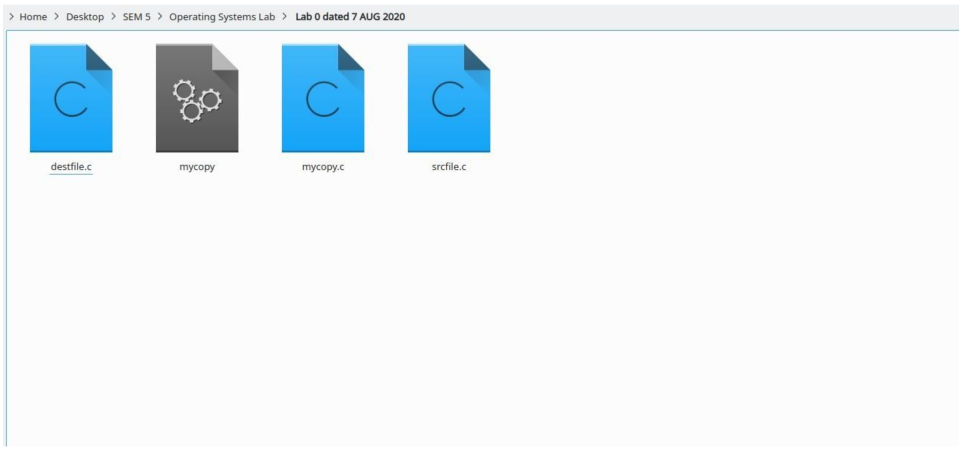


mycopy.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>

#define BUFF_SIZE 1024

int main(int argc, char* argv[])
{
    int srcFD, destFD, nbread, nbwrite;
    char *buff[BUFF_SIZE];

    if(argc == 2 && argv[1] == "--help")
    {
        printf("\nhelp feature is currently unavailable\n");
        exit(EXIT_FAILURE);
    }

    if(argc != 3)
    {
        printf("\nInsufficient number of arguments\n");
        exit(EXIT_FAILURE);
    }

    srcFD = open(argv[1], O_RDONLY);

    if(srcFD == -1)
    {
        printf("\nError opening file %s errno = %d\n", argv[1], errno);
        exit(EXIT_FAILURE);
```



```
    if(srcFD == -1)
    {
        printf("\nError opening file %s errno = %d\n", argv[1], errno);
        exit(EXIT_FAILURE);
    }

    destFD = open(argv[2], O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH);

    if(destFD == -1)
    {
        printf("\nError opening file %s errno = %d\n", argv[2], errno);
        exit(EXIT_FAILURE);
    }

    while((nbread = read(srcFD, buff, BUFF_SIZE)) > 0)
    {
        if(write(destFD, buff, nbread) != nbread)
            printf("\nError in writing data to %s\n",argv[2]);
    }

    if(nbread == -1)
        printf("\nError in writing data to %s\n", argv[1]);

    if(close(srcFD) == -1)
        printf("\nError in closing file %s\n",argv[1]);

    if(close(destFD) == -1)
        printf("\nError in closing file %s\n",argv[2]);

    exit(EXIT_SUCCESS);
}
```

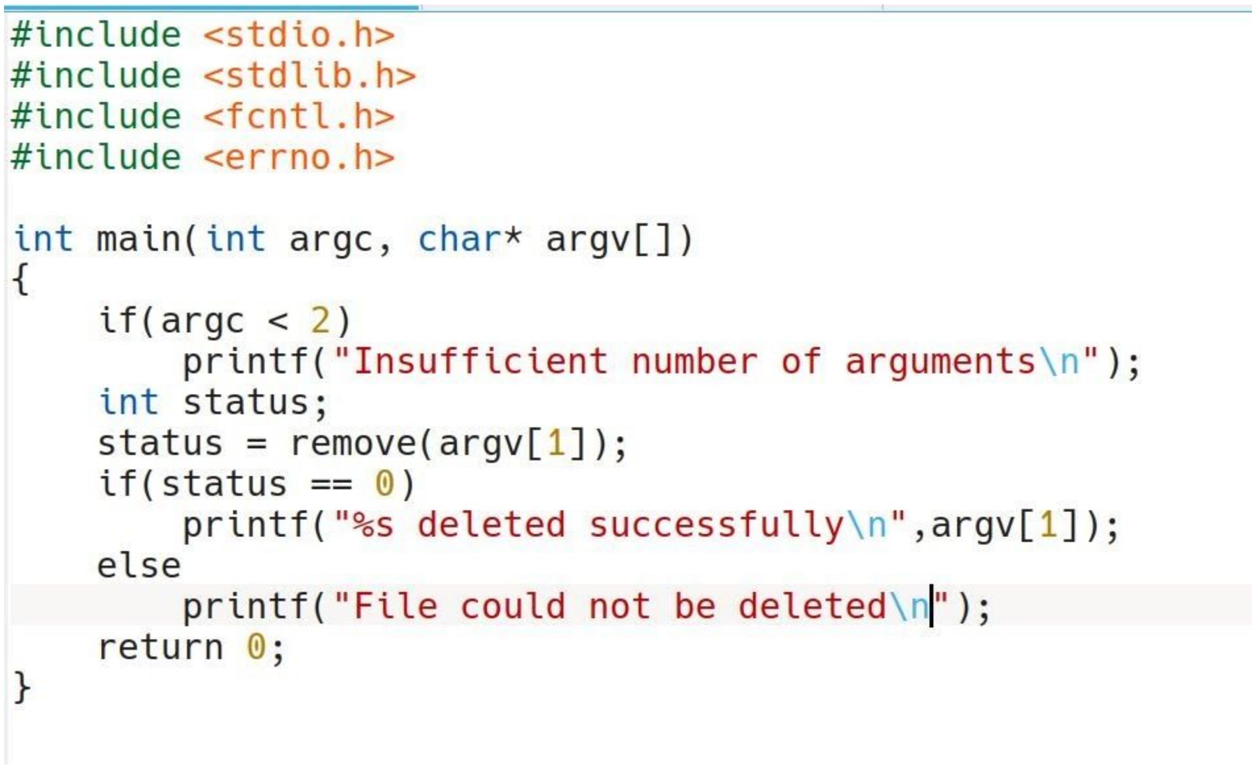

Home > Desktop > SEM 5 > Operating Systems Lab > Lab 0 dated 7 AUG 2020
destfile.c
mycopy
mycopy.c
srcfile.c

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



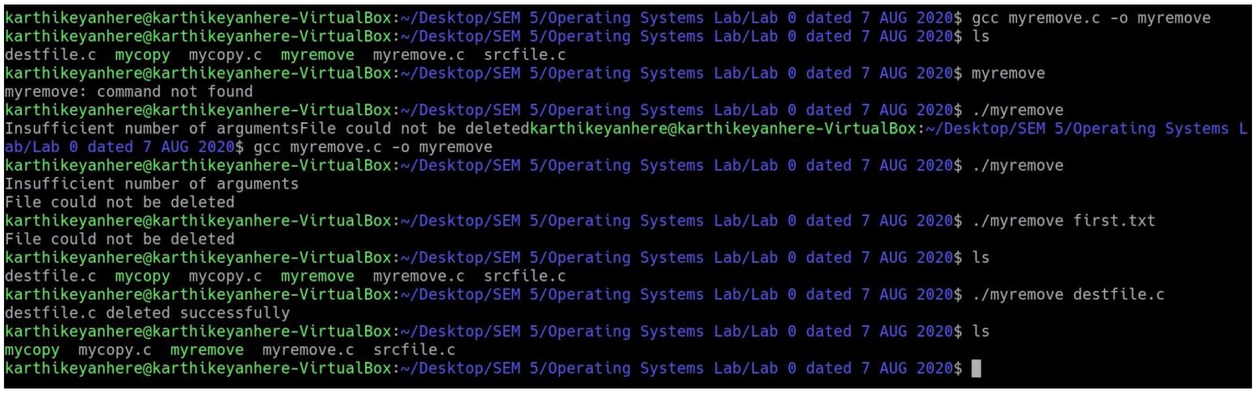
```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>

int main(int argc, char* argv[])
{
    if(argc < 2)
        printf("Insufficient number of arguments\n");
    int status;
    status = remove(argv[1]);
    if(status == 0)
        printf("%s deleted successfully\n",argv[1]);
    else
        printf("File could not be deleted\n");
    return 0;
}
```

COMMANDS

terminal~ ./myremove destfile.c
destfile.c deleted successfully

Terminal~ ./myremove
Insufficient number of arguments
File could not be deleted

OUTPUT



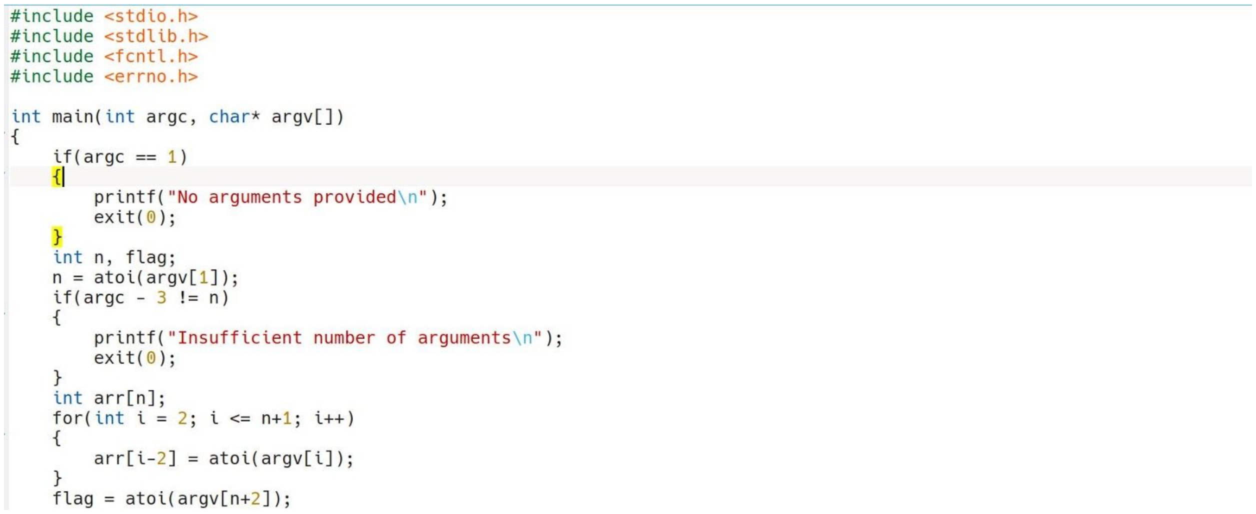
```
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ gcc myremove.c -o myremove
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ls
destfile.c  mycopy  mycopy.c  myremove  myremove.c  srcfile.c
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ myremove
myremove: command not found
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ./myremove
Insufficient number of argumentsFile could not be deletedkarthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems L
ab/Lab 0 dated 7 AUG 2020$ gcc myremove.c -o myremove
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ./myremove
Insufficient number of arguments
File could not be deleted
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ./myremove first.txt
File could not be deleted
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ls
destfile.c  mycopy  mycopy.c  myremove  myremove.c  srcfile.c
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ./myremove destfile.c
destfile.c deleted successfully
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ls
mycopy  mycopy.c  myremove  myremove.c  srcfile.c
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ 
```

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!

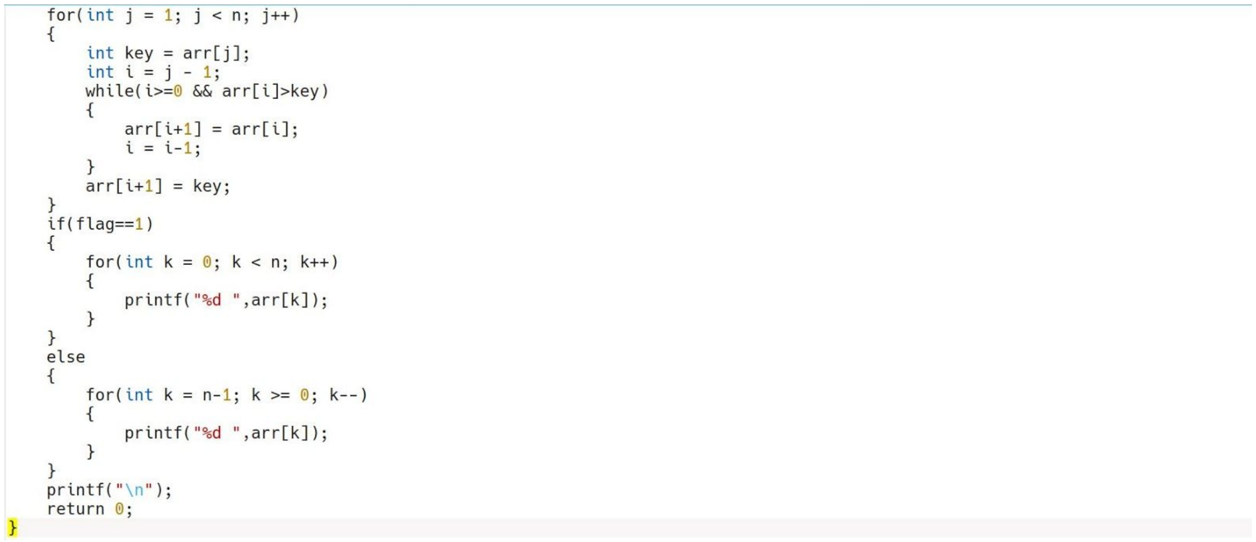
LOGIC

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



```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>

int main(int argc, char* argv[])
{
    if(argc == 1)
    {
        printf("No arguments provided\n");
        exit(0);
    }
    int n, flag;
    n = atoi(argv[1]);
    if(argc - 3 != n)
    {
        printf("Insufficient number of arguments\n");
        exit(0);
    }
    int arr[n];
    for(int i = 2; i <= n+1; i++)
    {
        arr[i-2] = atoi(argv[i]);
    }
    flag = atoi(argv[n+2]);
```

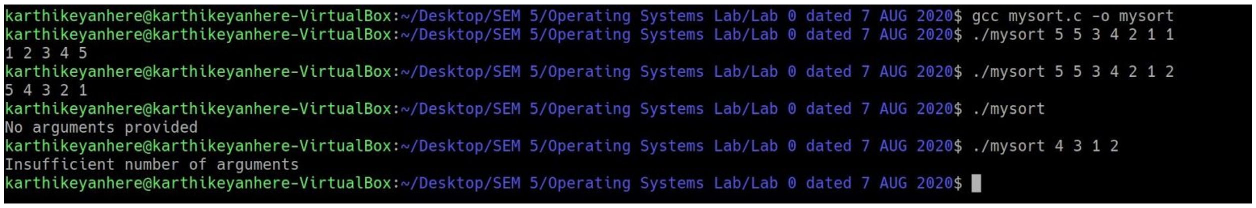


```c
    for(int j = 1; j < n; j++)
    {
        int key = arr[j];
        int i = j - 1;
        while(i>=0 && arr[i]>key)
        {
            arr[i+1] = arr[i];
            i = i-1;
        }
        arr[i+1] = key;
    }
    if(flag==1)
    {
        for(int k = 0; k < n; k++)
        {
            printf("%d ",arr[k]);
        }
    }
    else
    {
        for(int k = n-1; k >= 0; k--)
        {
            printf("%d ",arr[k]);
        }
    }
    printf("\n");
    return 0;
}
```

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



```
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ gcc mysort.c -o mysort
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ./mysort 5 5 3 4 2 1 1
1 2 3 4 5
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ./mysort 5 5 3 4 2 1 2
5 4 3 2 1
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ./mysort
No arguments provided
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ./mysort 4 3 1 2
Insufficient number of arguments
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ 
```

III) Develop an application (using function overloading & command line arguments in C) for:

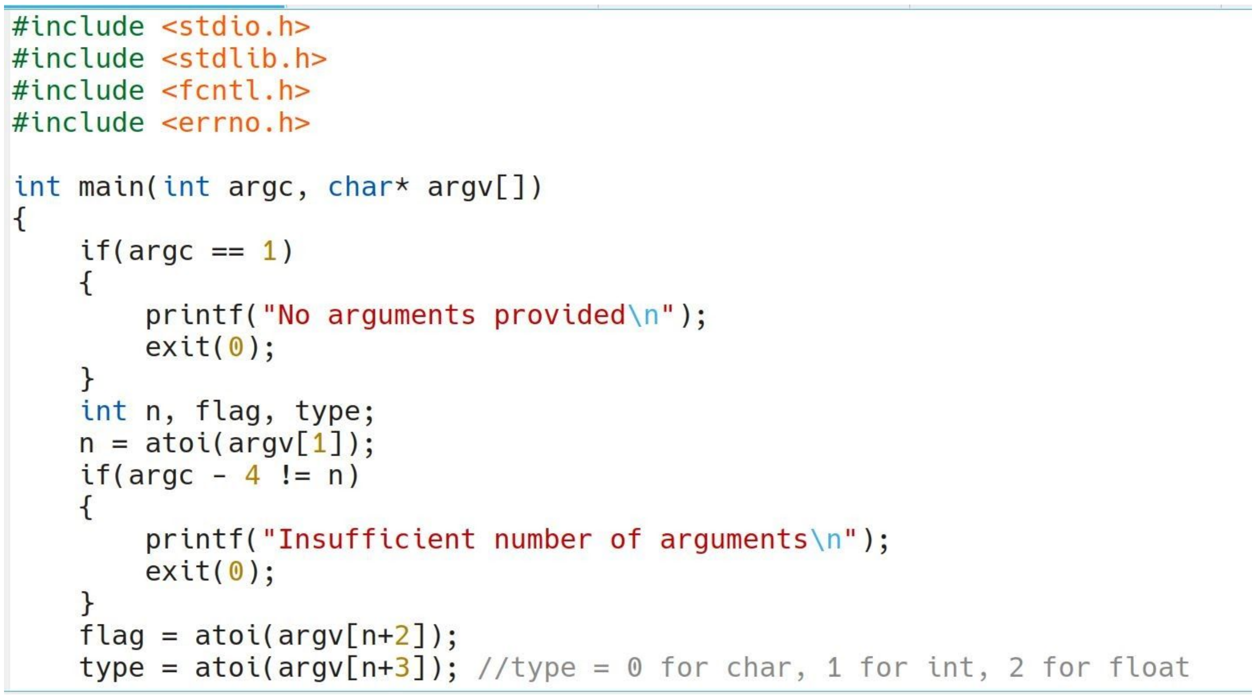
a) Sorting an array of integers or floating point or characters passed at command line. Usage syntax you can follow a similar style as for the II question and also support validation logic in the code.

LOGIC

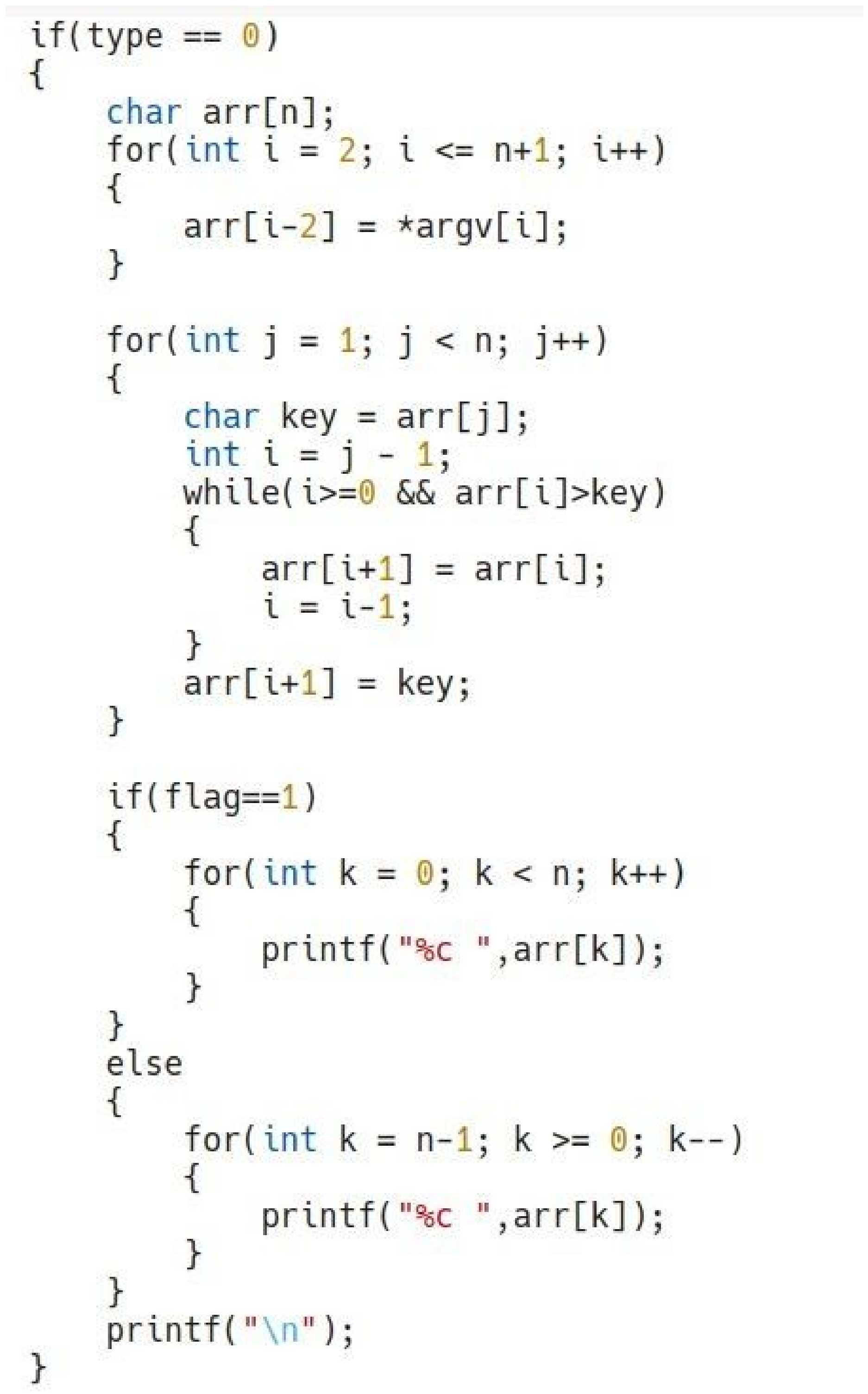
function overloading is implemented with the help of one extra parameter
ex ./mysortfo 5 b x r u w 1 0
In the above example, the last argument specifies that the elements of the array are characters, the second last argument (1 for ascending , else descending) specifies that sorting should be ascending.

0	character
1	integer
2	float

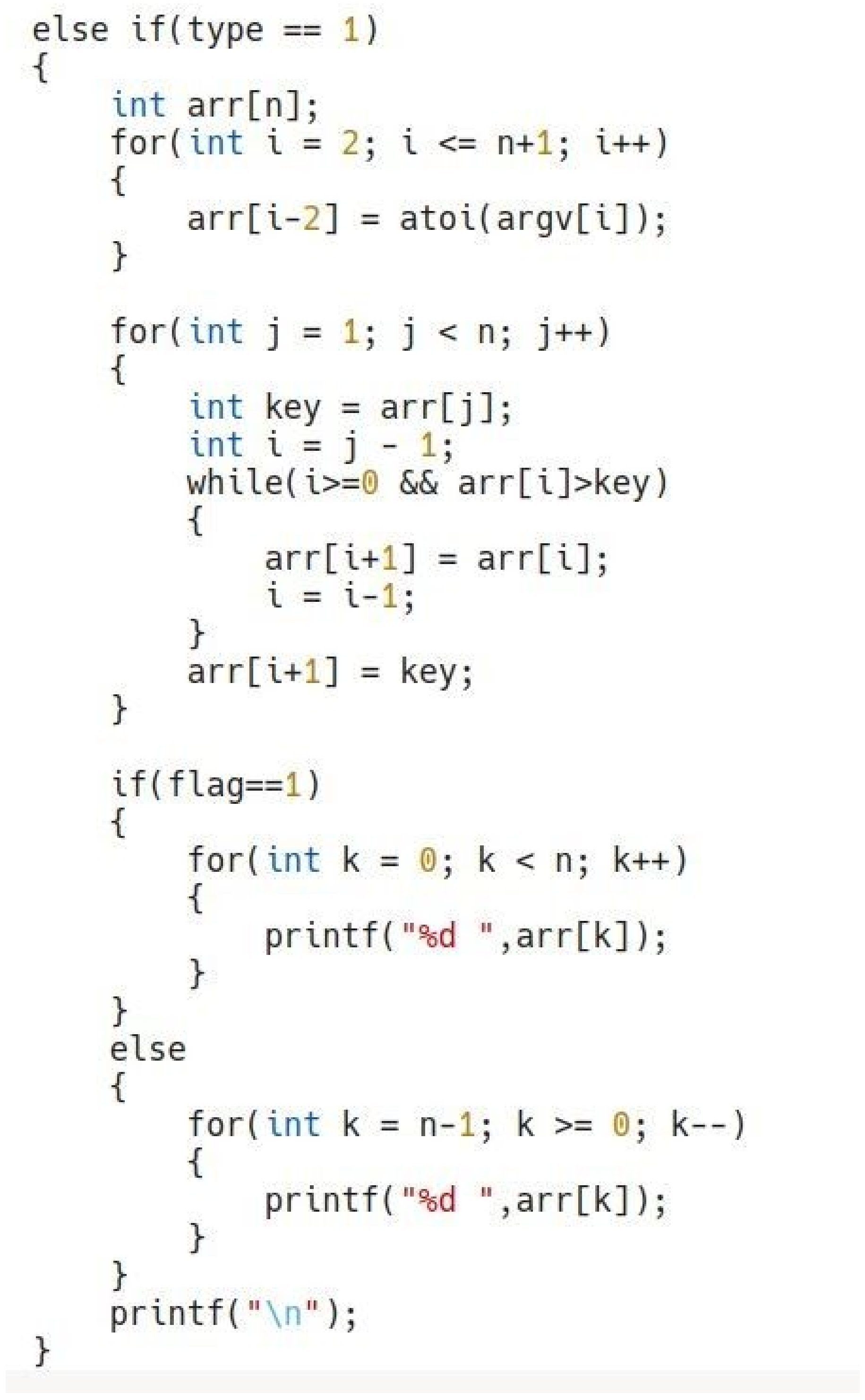
CODE



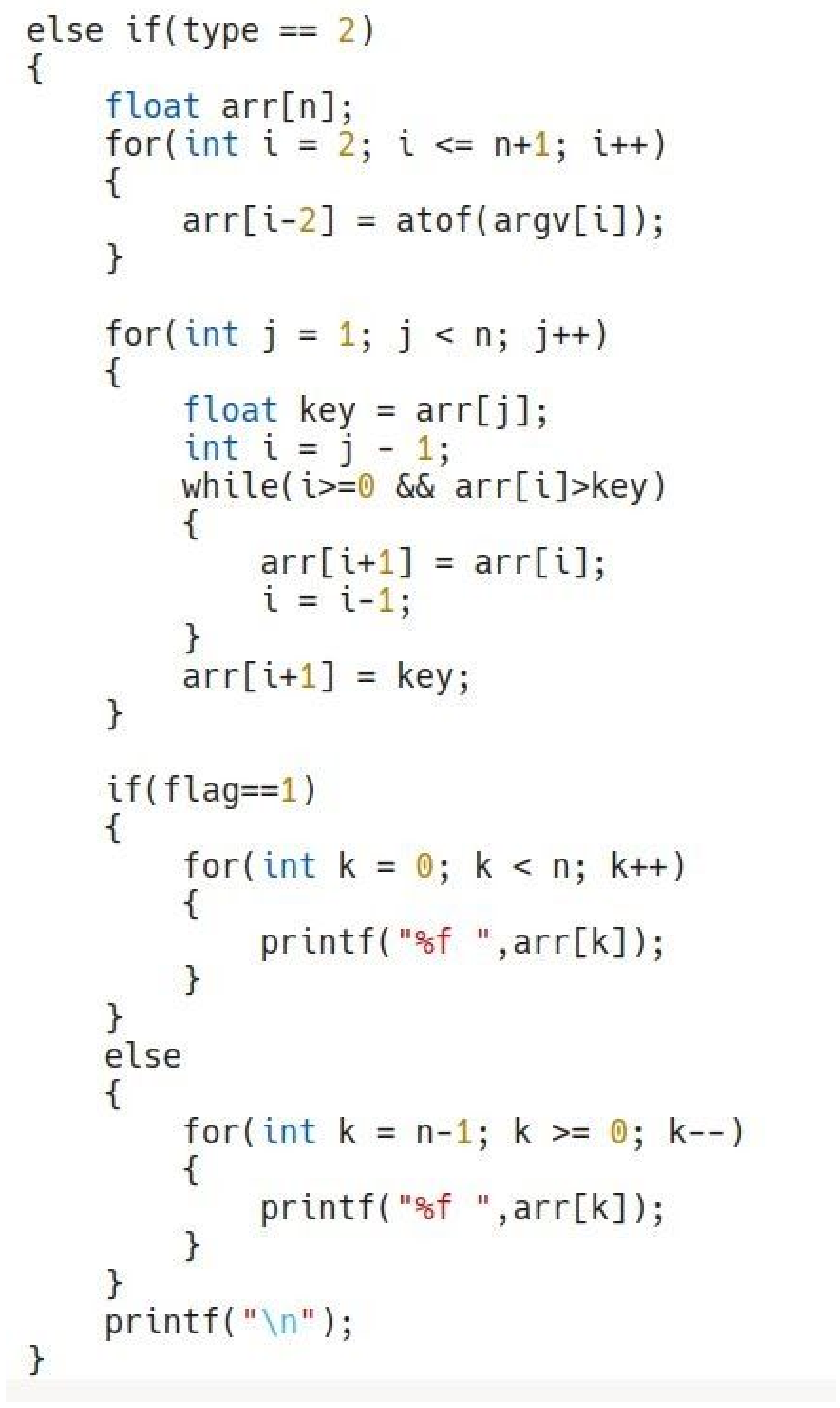
```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>

int main(int argc, char* argv[])
{
    if(argc == 1)
    {
        printf("No arguments provided\n");
        exit(0);
    }
    int n, flag, type;
    n = atoi(argv[1]);
    if(argc - 4 != n)
    {
        printf("Insufficient number of arguments\n");
        exit(0);
    }
    flag = atoi(argv[n+2]);
    type = atoi(argv[n+3]); //type = 0 for char, 1 for int, 2 for float
```



```c
if(type == 0)
{
    char arr[n];
    for(int i = 2; i <= n+1; i++)
    {
        arr[i-2] = *argv[i];
    }

    for(int j = 1; j < n; j++)
    {
        char key = arr[j];
        int i = j - 1;
        while(i>=0 && arr[i]>key)
        {
            arr[i+1] = arr[i];
            i = i-1;
        }
        arr[i+1] = key;
    }

    if(flag==1)
    {
        for(int k = 0; k < n; k++)
        {
            printf("%c ",arr[k]);
        }
    }
    else
    {
        for(int k = n-1; k >= 0; k--)
        {
            printf("%c ",arr[k]);
        }
    }
    printf("\n");
}
```



```c
else if(type == 1)
{
    int arr[n];
    for(int i = 2; i <= n+1; i++)
    {
        arr[i-2] = atoi(argv[i]);
    }

    for(int j = 1; j < n; j++)
    {
        int key = arr[j];
        int i = j - 1;
        while(i>=0 && arr[i]>key)
        {
            arr[i+1] = arr[i];
            i = i-1;
        }
        arr[i+1] = key;
    }

    if(flag==1)
    {
        for(int k = 0; k < n; k++)
        {
            printf("%d ",arr[k]);
        }
    }
    else
    {
        for(int k = n-1; k >= 0; k--)
        {
            printf("%d ",arr[k]);
        }
    }
    printf("\n");
}
```



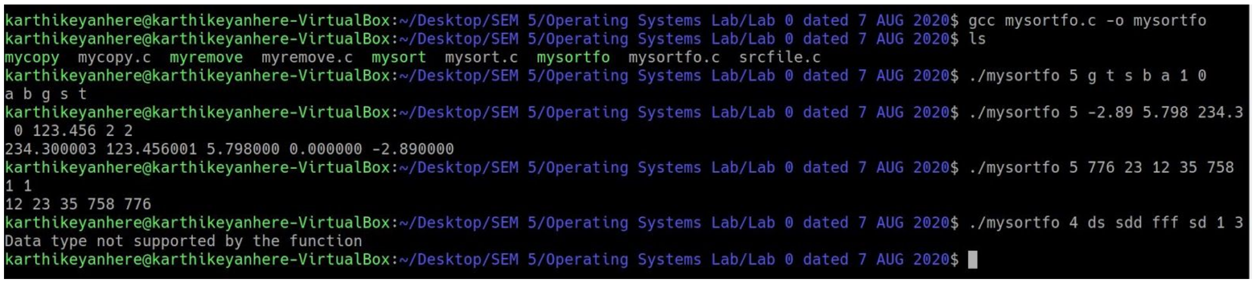
```c
else if(type == 2)
{
    float arr[n];
    for(int i = 2; i <= n+1; i++)
    {
        arr[i-2] = atof(argv[i]);
    }

    for(int j = 1; j < n; j++)
    {
        float key = arr[j];
        int i = j - 1;
        while(i>=0 && arr[i]>key)
        {
            arr[i+1] = arr[i];
            i = i-1;
        }
        arr[i+1] = key;
    }

    if(flag==1)
    {
        for(int k = 0; k < n; k++)
        {
            printf("%f ",arr[k]);
        }
    }
    else
    {
        for(int k = n-1; k >= 0; k--)
        {
            printf("%f ",arr[k]);
        }
    }
    printf("\n");
}
```

```
    else
    {
        printf("Data type not supported by the function\n");
        exit(0);
    }

    return 0;
}
```

COMMANDS

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

OUTPUT



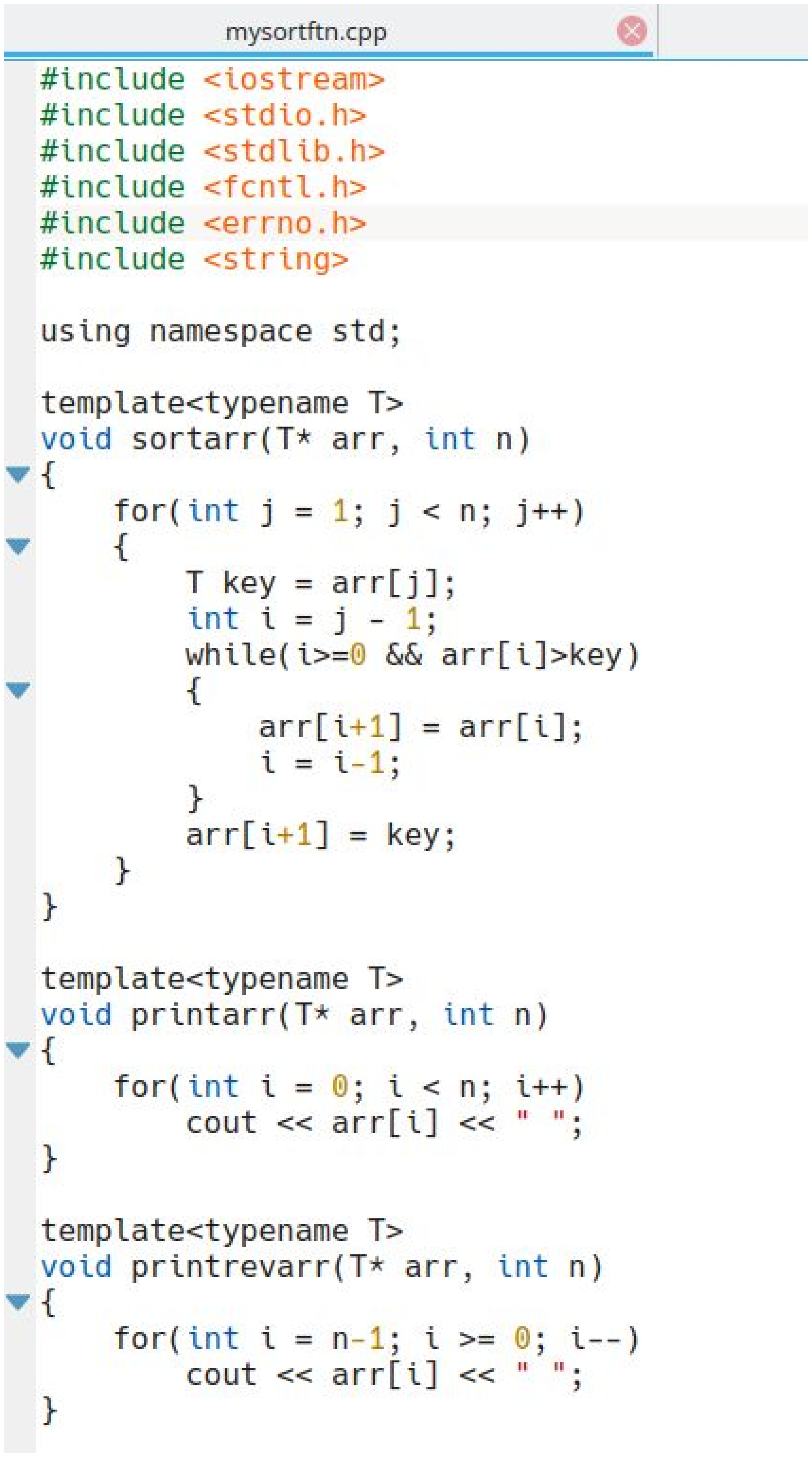
```
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ gcc mysortfo.c -o mysortfo
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ls
mycopy  mycopy.c  myremove  myremove.c  mysort  mysort.c  mysortfo  mysortfo.c  srcfile.c
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ./mysortfo 5 g t s b a 1 0
a b g s t
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ./mysortfo 5 -2.89 5.798 234.3
 0 123.456 2 2
234.300003 123.456001 5.798000 0.000000 -2.890000
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ./mysortfo 5 776 23 12 35 758
1 1
12 23 35 758 776
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ./mysortfo 4 ds sdd fff sd 1 3
Data type not supported by the function
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ █
```

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

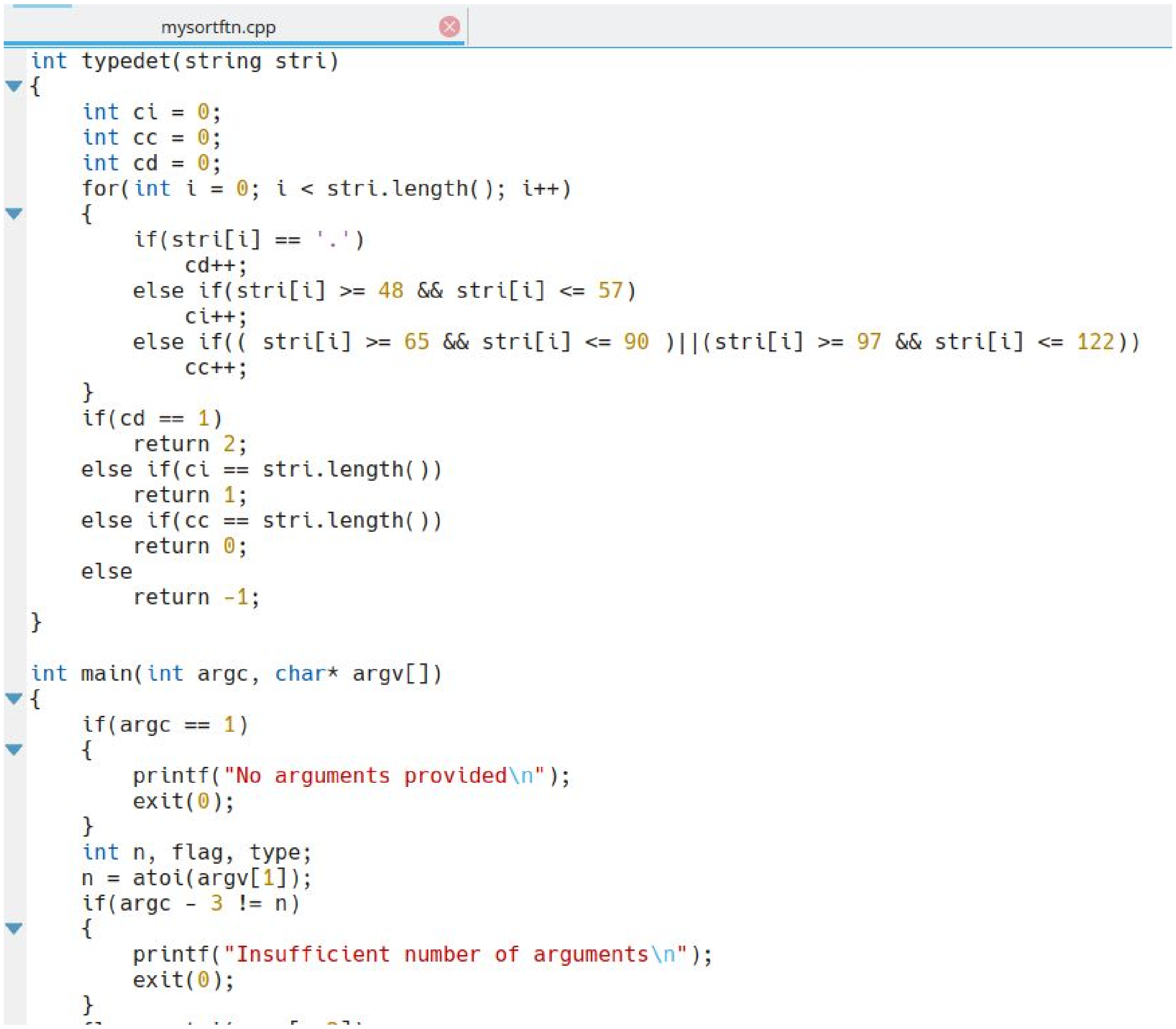
Function templates concept is implemented in C. A argv (string) is identified as a character array or integer or floating point number based on ascii values of argv(string).

CODE



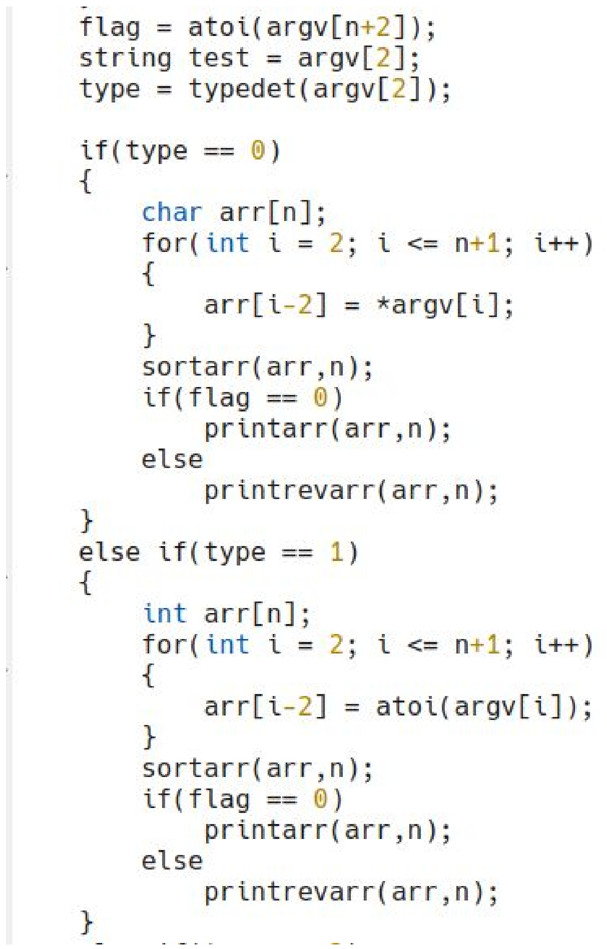
mysortftn.cpp

```cpp
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <string>

using namespace std;

template<typename T>
void sortarr(T* arr, int n)
{
    for(int j = 1; j < n; j++)
    {
        T key = arr[j];
        int i = j - 1;
        while(i>=0 && arr[i]>key)
        {
            arr[i+1] = arr[i];
            i = i-1;
        }
        arr[i+1] = key;
    }
}

template<typename T>
void printarr(T* arr, int n)
{
    for(int i = 0; i < n; i++)
        cout << arr[i] << " ";
}

template<typename T>
void printrevarr(T* arr, int n)
{
    for(int i = n-1; i >= 0; i--)
        cout << arr[i] << " ";
}
```

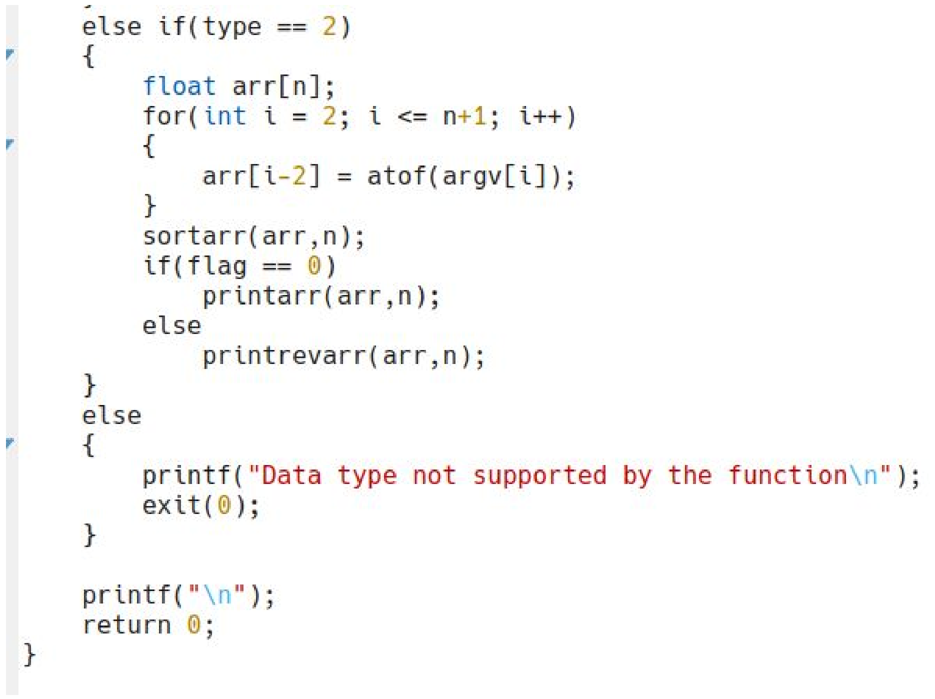


mysortftn.cpp

```cpp
int typedet(string stri)
{
    int ci = 0;
    int cc = 0;
    int cd = 0;
    for(int i = 0; i < stri.length(); i++)
    {
        if(stri[i] == '.')
            cd++;
        else if(stri[i] >= 48 && stri[i] <= 57)
            ci++;
        else if(( stri[i] >= 65 && stri[i] <= 90 )||(stri[i] >= 97 && stri[i] <= 122))
            cc++;
    }
    if(cd == 1)
        return 2;
    else if(ci == stri.length())
        return 1;
    else if(cc == stri.length())
        return 0;
    else
        return -1;
}

int main(int argc, char* argv[])
{
    if(argc == 1)
    {
        printf("No arguments provided\n");
        exit(0);
    }
    int n, flag, type;
    n = atoi(argv[1]);
    if(argc - 3 != n)
    {
        printf("Insufficient number of arguments\n");
        exit(0);
    }
```



```
    flag = atoi(argv[n+2]);
    string test = argv[2];
    type = typedet(argv[2]);

    if(type == 0)
    {
        char arr[n];
        for(int i = 2; i <= n+1; i++)
        {
            arr[i-2] = *argv[i];
        }
        sortarr(arr,n);
        if(flag == 0)
            printarr(arr,n);
        else
            printrevarr(arr,n);
    }
    else if(type == 1)
    {
        int arr[n];
        for(int i = 2; i <= n+1; i++)
        {
            arr[i-2] = atoi(argv[i]);
        }
        sortarr(arr,n);
        if(flag == 0)
            printarr(arr,n);
        else
            printrevarr(arr,n);
    }
```

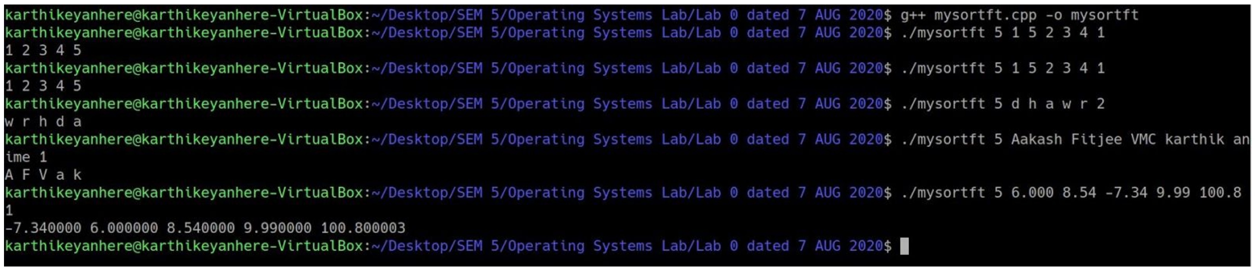


```
    else if(type == 2)
    {
        float arr[n];
        for(int i = 2; i <= n+1; i++)
        {
            arr[i-2] = atof(argv[i]);
        }
        sortarr(arr,n);
        if(flag == 0)
            printarr(arr,n);
        else
            printrevarr(arr,n);
    }
    else
    {
        printf("Data type not supported by the function\n");
        exit(0);
    }

    printf("\n");
    return 0;
}
```

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



```
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ g++ mysortft.cpp -o mysortft
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ./mysortft 5 1 5 2 3 4 1
1 2 3 4 5
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ./mysortft 5 1 5 2 3 4 1
1 2 3 4 5
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ./mysortft 5 d h a w r 2
w r h d a
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ./mysortft 5 Aakash Fitjee VMC karthik an
ime 1
A F V a k
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ ./mysortft 5 6.000 8.54 -7.34 9.99 100.8
1
-7.340000 6.000000 8.540000 9.990000 100.800003
karthikeyanhere@karthikeyanhere-VirtualBox:~/Desktop/SEM 5/Operating Systems Lab/Lab 0 dated 7 AUG 2020$ 
```