

OPERATING SYSTEMS(COM301T)- PROGRAMMING PREP-ASSIGNMENT

l) 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.

Approach: The approach includes basic concepts of file handling where the source is opened in read mode and the contents are dumped to another destination file which is opened with a file pointer in enabled write mode. Since the destination file is opened in write mode, if the user gives a filename that is not already present in the system, the program will create a new file.

Linux overview: cp stands for copy. This command is used to copy files. It creates an exact image of a file on a disk with a different file name.

Source code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char* argv[])
{
    char ch;
    int count = 0;

    /*Error if source file is not provided*/
    if(argc == 1)
    {
        printf("mycopy: missing file operand\n");
        exit(EXIT_FAILURE);
    }

    /*Error if source is provided but destination is not provided*/
    if(argc == 2)
    {
        printf("mycopy: missing destination file operand after %s\n", argv[1]);
        exit(EXIT_FAILURE);
    }

    /*Error if too many arguments are passed*/
```

```

if(argc > 3)
{
    printf("Too many arguments passed!\n");
    exit(EXIT_FAILURE);
}

/*Open source file in read mode*/
FILE *source;
source = fopen(argv[1], "rb");

/*If source file is not present in that directory, give error*/
if(source == NULL)
{
    printf("\nmycopy: cannot stat %s: No such file or directory\n", argv[1]);
    exit(EXIT_FAILURE);
}

/*Open destination file in write mode*/
FILE *destination;
destination = fopen(argv[2], "wb");

/*count the number of characters in source file*/
while((ch=fgetc(source))!=EOF)
{
    count++;
}

/*Create a buffer that will hold the content that needs to be copied*/
char buffer[count];

/*Position the file pointer at the starting of the source file*/
fseek(source, 0, SEEK_SET);

/*Transfer data from source file to destination file*/
fread(buffer, count, 1, source);
fwrite(buffer, count, 1, destination);
printf("\nSuccessfully copied\n");

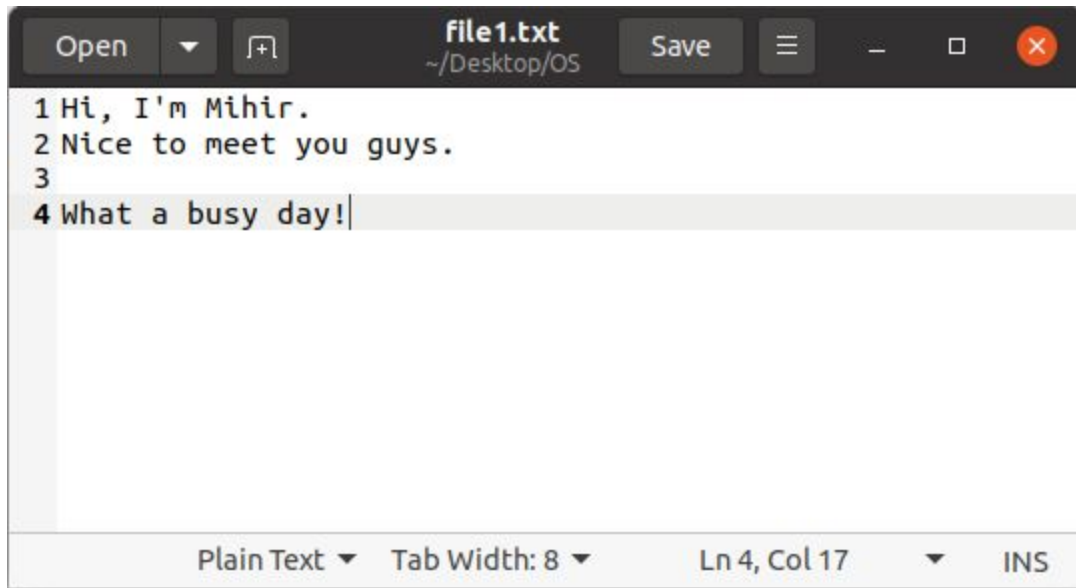
/*Close the file*/
fclose(source);

exit(EXIT_SUCCESS);
}

```

Input screenshots:

1. file1.txt

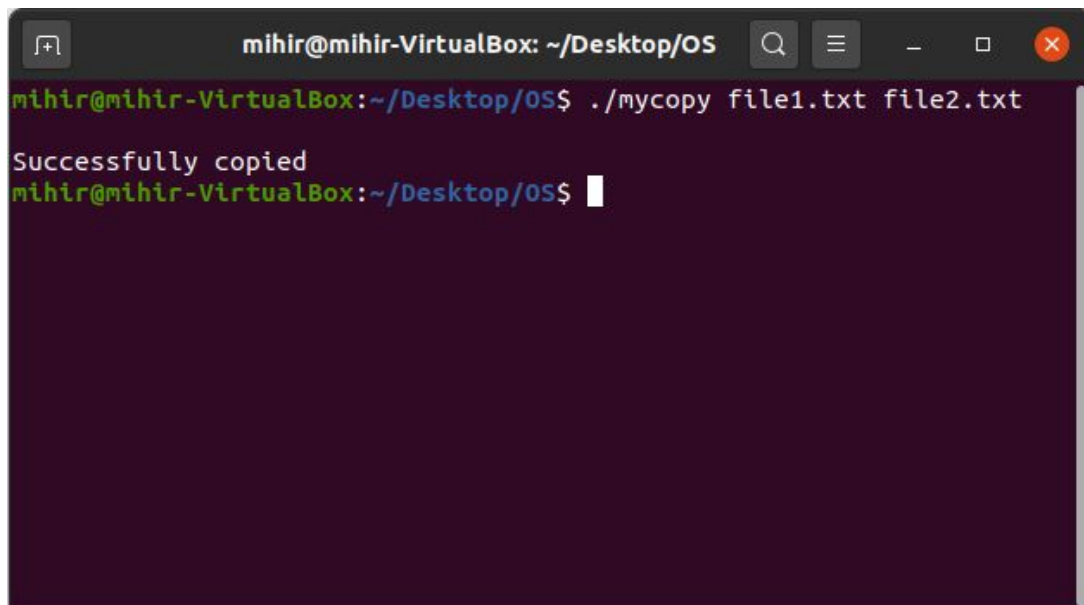


A screenshot of a text editor window titled "file1.txt" with the path "~/Desktop/OS". The window has a dark theme and includes buttons for "Open", "Save", and window controls. The text content is as follows:

```
1 Hi, I'm Mihir.  
2 Nice to meet you guys.  
3  
4 What a busy day!
```

The status bar at the bottom indicates "Plain Text", "Tab Width: 8", "Ln 4, Col 17", and "INS" mode.

2. Terminal



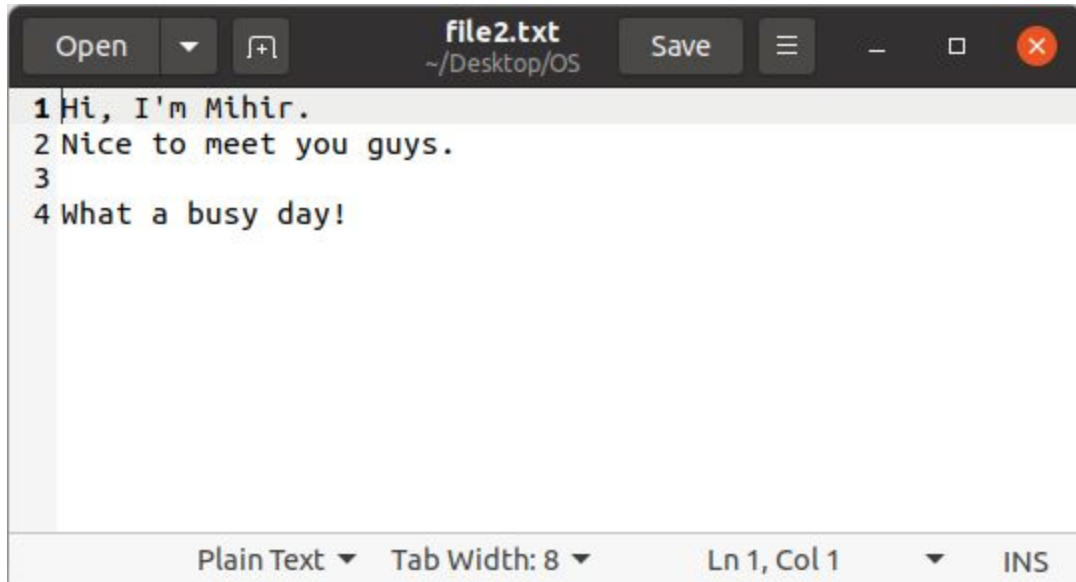
A screenshot of a terminal window titled "mihir@mihir-VirtualBox: ~/Desktop/OS". The terminal shows the execution of a command to copy a file:

```
mihir@mihir-VirtualBox:~/Desktop/OS$ ./mycopy file1.txt file2.txt  
Successfully copied  
mihir@mihir-VirtualBox:~/Desktop/OS$
```

The terminal has a dark purple background and a light-colored cursor.

Output Screenshots:

1. file2.txt (newly created since it was not initially present)



```
file2.txt
~/Desktop/OS
Open Save
1 Hi, I'm Mihir.
2 Nice to meet you guys.
3
4 What a busy day!
Plain Text Tab Width: 8 Ln 1, Col 1 INS
```

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.

Approach: In the C Programming Language, the remove function removes a file pointed to by filename. The function is used to delete a file.

Source code:

```
#include <stdio.h>
#include <stdlib.h>
```

```
int main(int argc, char* argv[])
{
    int status;
    char file_name[50];
```

```

/*Error if file name is not provided*/
if (argc == 1)
{
    printf("myrm: missing operand");
    exit(EXIT_FAILURE);
}

/*Error if more than 2 arguments passed*/
if (argc > 2)
{
    printf("myrm: Too many arguments!");
    exit(EXIT_FAILURE);
}

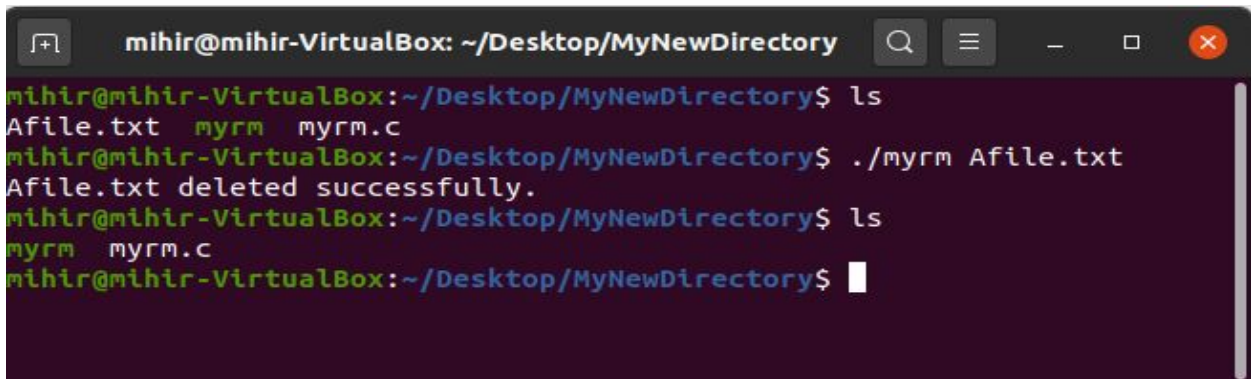
for (int i=1; i<argc; i++)
{
    /*Delete the file*/
    status = remove(argv[i]);

    /*Check if deletion was successful*/
    if (status == 0)
        printf("%s deleted successfully.\n", argv[i]);
    else
    {
        printf("Unable to delete the file\n");
        perror("Error\n");
    }
}

return 0;
}

```

Output screenshot:



```

mihir@mihir-VirtualBox: ~/Desktop/MyNewDirectory
mihir@mihir-VirtualBox:~/Desktop/MyNewDirectory$ ls
Afile.txt  myrm  myrm.c
mihir@mihir-VirtualBox:~/Desktop/MyNewDirectory$ ./myrm Afile.txt
Afile.txt deleted successfully.
mihir@mihir-VirtualBox:~/Desktop/MyNewDirectory$ ls
myrm  myrm.c
mihir@mihir-VirtualBox:~/Desktop/MyNewDirectory$

```

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.

Approach: The main function takes command line arguments and asks the user to input the size of the array, the ascending/descending choice and the elements of the array. The elements are entered as characters hence they are converted to integers using strtoint() function, the definition of which is programmed. The array is sorted depending on the user's choice.

Linux overview: In computing, sort is a standard command line program of Unix-like operating systems, that prints the lines of its input or concatenation of all files listed in its argument list in sorted order. Sorting is done based on one or more sort keys extracted from each line of input.

Source code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
/*A function to swap two integers*/
void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}
```

```
/*Bubble sort to sort in ascending order*/
void bubbleSortAsc(int arr[], int n)
{
    int i, j;
    for (i = 0; i < n-1; i++)

        // Last i elements are already in place
        for (j = 0; j < n-i-1; j++)
```

```

        if (arr[j] > arr[j+1])
            swap(&arr[j], &arr[j+1]);
    }

```

```

/*A function to print and array*/
void printArray(int arr[], int size)
{
    int i;
    printf("Sorted array:\n");
    for (i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

```

```

/*Bubble sort to sort in descending order*/
void bubbleSortDsc(int arr[], int n)
{
    int i, j;
    for (i = 0; i < n-1; i++)

        // Last i elements are already in place
        for (j = 0; j < n-i-1; j++)
            if (arr[j] < arr[j+1])
                swap(&arr[j], &arr[j+1]);
}

```

```

int main(int argc, char* argv[])
{
    /*Convert the required command line arguments to integer*/
    int size = atoi(argv[1]), flag = atoi(argv[2]);
    int a[size];

    /*Convert the array elements from string to integer*/
    for (int i = 0; i < size; i++)
    {
        a[i] = atoi(argv[i+3]);
    }

    if (flag == 1)
        bubbleSortAsc(a, size);
}

```

```

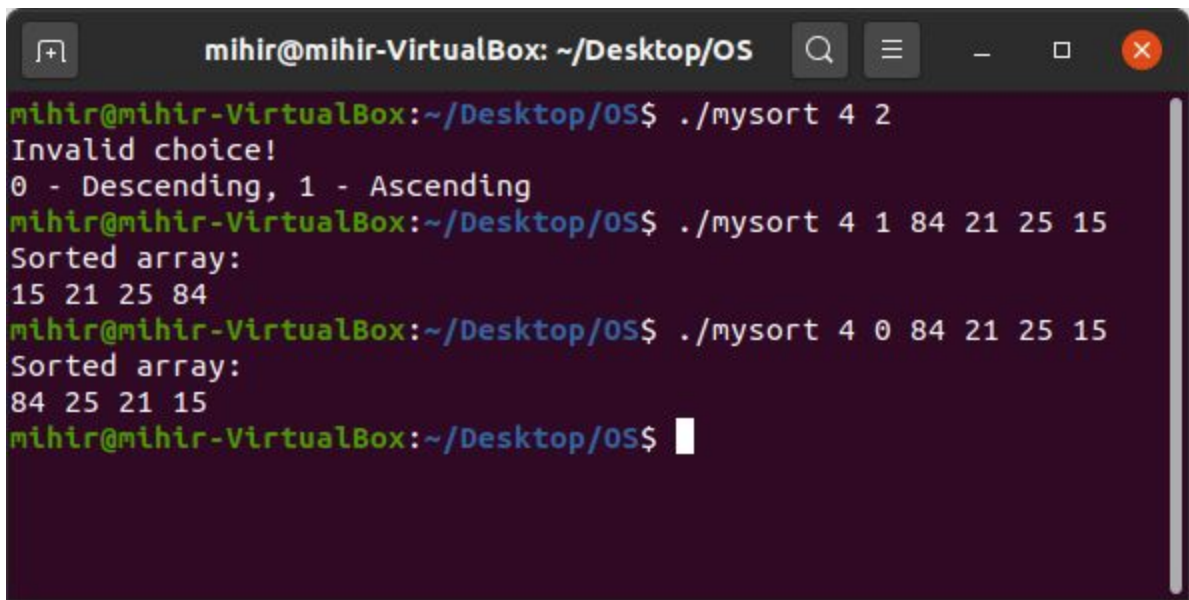
        else if (flag == 2)
            bubbleSortDsc(a, size);

        printArray(a, size);

        return 0;
    }

```

Output screenshot:



The screenshot shows a terminal window titled "mihir@mihir-VirtualBox: ~/Desktop/OS". The user runs the command `./mysort 4 2`, which outputs "Invalid choice!" and "0 - Descending, 1 - Ascending". Then, the user runs `./mysort 4 1 84 21 25 15`, which outputs "Sorted array:" followed by "15 21 25 84". Finally, the user runs `./mysort 4 0 84 21 25 15`, which outputs "Sorted array:" followed by "84 25 21 15". The terminal window has a dark purple background and standard window controls at the top.

```

mihir@mihir-VirtualBox: ~/Desktop/OS$ ./mysort 4 2
Invalid choice!
0 - Descending, 1 - Ascending
mihir@mihir-VirtualBox: ~/Desktop/OS$ ./mysort 4 1 84 21 25 15
Sorted array:
15 21 25 84
mihir@mihir-VirtualBox: ~/Desktop/OS$ ./mysort 4 0 84 21 25 15
Sorted array:
84 25 21 15
mihir@mihir-VirtualBox: ~/Desktop/OS$

```

Extra Credits Qn:

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!

Approach: Function Pointers are just like normal pointers with the special ability of being able to point to a function rather than a variable. In Functions Pointers, function's name can be used to get the function's address. A function can also be passed as an argument and can be

returned from a function. If we observe carefully, then the only difference between bubble sort for ascending order and bubble sort for descending order is in the line where the adjacent elements are being compared to each other. So, we can create two different functions which return 1 if $\text{num1} < \text{num2}$ and $\text{num1} > \text{num2}$ respectively and then use function pointers to call them as and when needed.

Source code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/*A function to print and array*/
void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

// A function which returns 1 if a is smaller than b
int less_than(int a, int b)
{
    return a < b;
}

// A function which returns 1 if a is greater than b
int greater_than(int a,int b)
{
    return a > b;
}

// A function for bubble sort algorithm with function pointer
void bubbleSort(int arr[], int n, int flag)
{
    int (*asc_or_dsc[])(int, int) = {less_than, greater_than};
    int i, j;
    for (i = 0; i < n-1; i++)
        for (j = 0; j < n-i-1; j++)
            if((*asc_or_dsc[flag])(arr[j], arr[j+1]))
                swap(&arr[j], &arr[j+1]);
}
```

```
}
```

```
// A function to print an array
```

```
void printArray(int arr[], int size)
```

```
{
```

```
    int i;
```

```
    printf("Sorted array:\n");
```

```
    for (i = 0; i < size; i++)
```

```
        printf("%d ", arr[i]);
```

```
    printf("\n");
```

```
}
```

```
int main(int argc, char* argv[])
```

```
{
```

```
    // Convert the required command line arguments from string to integer
```

```
    int size = atoi(argv[1]), flag = atoi(argv[2]);
```

```
    int a[size];
```

```
    if (size <= 0)
```

```
    {
```

```
        printf("Invalid size!\nPlease enter size > 0\n");
```

```
        exit(EXIT_FAILURE);
```

```
    }
```

```
    if (flag > 1)
```

```
    {
```

```
        printf("Invalid choice!\n0 - Descending, 1 - Ascending\n");
```

```
        exit(EXIT_FAILURE);
```

```
    }
```

```
    for (int i = 0; i < size; i++)
```

```
    {
```

```
        a[i] = atoi(argv[i+3]);
```

```
    }
```

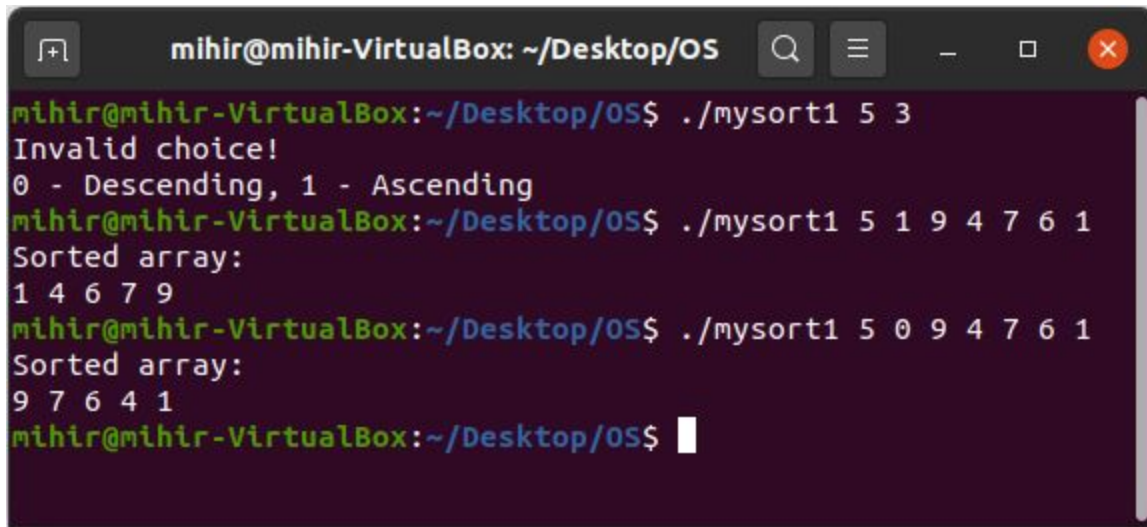
```
    bubbleSort(a, size, flag);
```

```
    printArray(a, size);
```

```
    return 0;
```

```
}
```

Output screenshot:

A screenshot of a terminal window titled 'mihir@mihir-VirtualBox: ~/Desktop/OS'. The terminal shows the execution of a program named 'mysort1'. The user enters './mysort1 5 3', and the program outputs 'Invalid choice!'. The user then enters './mysort1 5 1 9 4 7 6 1', and the program outputs 'Sorted array: 1 4 6 7 9'. Finally, the user enters './mysort1 5 0 9 4 7 6 1', and the program outputs 'Sorted array: 9 7 6 4 1'. The terminal window has a dark background and standard window controls at the top.

```
mihir@mihir-VirtualBox: ~/Desktop/OS
mihir@mihir-VirtualBox:~/Desktop/OS$ ./mysort1 5 3
Invalid choice!
0 - Descending, 1 - Ascending
mihir@mihir-VirtualBox:~/Desktop/OS$ ./mysort1 5 1 9 4 7 6 1
Sorted array:
1 4 6 7 9
mihir@mihir-VirtualBox:~/Desktop/OS$ ./mysort1 5 0 9 4 7 6 1
Sorted array:
9 7 6 4 1
mihir@mihir-VirtualBox:~/Desktop/OS$
```

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.

Approach: In addition to the approach in question 2, the logic now accepts input from integer, char, and floating data types from the user. Three functions with the same name bubbleSort() are used to handle the data types using the concept of overloading, the only difference in the signatures being the accepting the type of array. In some programming languages, function overloading or method overloading is the ability to create multiple functions of the same name with different implementations.

Source code:

```
#include<unistd.h>
#include<stdio.h>
#include <stdlib.h>
#include <string.h>
#include<ctype.h>
```

```
using namespace std;
```

// A function which returns 1 if integer a is smaller than integer b

```
int less_than(int a, int b)
{
    return a < b;
}
```

// A function which returns 1 if integer a is greater than integer b

```
int greater_than(int a,int b)
{
    return a > b;
}
```

// A function which returns 1 if float a is smaller than float b

```
int less_than(float a, float b)
{
    return a < b;
}
```

// A function which returns 1 if float a is greater than float b

```
int greater_than(float a, float b)
{
    return a > b;
}
```

// Bubble sort function which accepts integer array

```
void bubbleSort(int arr[], int n, int flag)
{
    int (*asc_or_dsc[])(int, int) = {less_than, greater_than};
    int i, j;
    for (i = 0; i < n-1; i++)
        for (j = 0; j < n-i-1; j++)
            if ((*asc_or_dsc[flag])(arr[j], arr[j+1]))
            {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
    printf("Sorted array:\n");
}
```

```

        for (i = 0; i < n; i++)
            printf("%d ", arr[i]);
        printf("\n");
    }

```

// Bubble sort function which accepts float array

```

void bubbleSort(double *arr, int n, int flag)
{
    int (*asc_or_dsc[])(float, float) = {less_than, greater_than};
    int i, j;
    for (i = 0; i < n-1; i++)
        for (j = 0; j < n-i-1; j++)
            if ((*asc_or_dsc[flag])(arr[j], arr[j+1]))
            {
                double temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }

    printf("Sorted array:\n");
    for (i = 0; i < n; i++)
        printf("%f ", arr[i]);
    printf("\n");
}

```

// Bubble sort function which accepts character array

```

void bubbleSort(char *arr, int n, int flag)
{
    int (*asc_or_dsc[])(int, int) = {less_than, greater_than};
    int i, j;
    for (i = 0; i < n-1; i++)
        for (j = 0; j < n-i-1; j++)
            if ((*asc_or_dsc[flag])(arr[j], arr[j+1]))
            {
                char temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
}

```

```
// A function to check if the input character is an integer or a floating point number
```

```
int isInteger(char* str)
{
    char character;
    for(int i=0;i<strlen(str);++i)
    {
        character = str[i];
        if(character == '.')
            return 0;
    }
    return 1;
}
```

```
int main(int argc, char* argv[])
```

```
{
    // Convert the required command line arguments from string to integer
    int size = atoi(argv[1]), flag = atoi(argv[2]);
    int integer = 0;
    char a[size];
    int b[size];
    double c[size];

    if (size <= 0)
    {
        printf("Invalid size!\nPlease enter size > 0\n");
        exit(EXIT_FAILURE);
    }

    if (flag > 1)
    {
        printf("Invalid choice!\n0 - Descending, 1 - Ascending\n");
        exit(EXIT_FAILURE);
    }

    // Check if the first element of the array is an alphabet or not
    if (isalpha(argv[3][0]))
    {
        for (int i = 0; i < size; i++)
        {
            a[i] = argv[i+3][0];
        }
    }
}
```

```

    }
    bubbleSort(a, size, flag);
    printf("Sorted array:\n");
        for (int i = 0; i < size; i++)
            printf("%c ", a[i]);
            printf("\n");
    }

else
{
    for (int i = 0; i < size; i++)
    {
        // Check if all the elements of the array are integers.
        // Even if a single element is of float type, the array qualifies as a float array
        if (!isInteger(argv[i+3]))
        {
            integer = 1;
            break;
        }
    }

    if (integer == 0)
    {
        for (int i = 0; i < size; i++)
        {
            b[i] = atoi(argv[i+3]);
        }

        bubbleSort(b, size, flag);
    }

    else
    {
        for (int i = 0; i < size; i++)
        {
            c[i] = atof(argv[i+3]);
        }

        bubbleSort(c, size, flag);
    }
}
}

```

```
    return 0;  
}
```

Output screenshot:



```
mihir@mihir-VirtualBox: ~/Desktop/OS$ ./FuncOverload 5 4  
Invalid choice!  
0 - Descending, 1 - Ascending  
mihir@mihir-VirtualBox: ~/Desktop/OS$ ./FuncOverload 5 1 5 8 2 1 3  
Sorted array:  
1 2 3 5 8  
mihir@mihir-VirtualBox: ~/Desktop/OS$ ./FuncOverload 5 0 5.4 5.8 2.6 5.5 6.9  
Sorted array:  
6.900000 5.800000 5.500000 5.400000 2.600000  
mihir@mihir-VirtualBox: ~/Desktop/OS$ ./FuncOverload 5 1 j g z a b  
Sorted array:  
a b g j z  
mihir@mihir-VirtualBox: ~/Desktop/OS$
```

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!

Approach: Templates are a feature of the C++ programming language that allows functions and classes to operate with generic types. This allows a function or class to work on many different data types without being rewritten for each one. The program uses a single sort function with a datatype T templated for use.

Source code:

```
#include <bits/stdc++.h>  
#include <unistd.h>  
#include <iostream>  
#include <stdio.h>
```



```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
using namespace std;
```

```
// A function which returns 1 if an element a of datatype T is smaller than an element b of
datatype T
```

```
template<typename T>
int less_than(T a, T b)
{
    return a < b;
}
```

```
// A function which returns 1 if an element a of datatype T is greater than an element b of
datatype T
```

```
template<typename T>
int greater_than(T a, T b)
{
    return a > b;
}
```

```
// A bubble sort function which accepts an array of data type T
```

```
template<typename T>
void bubbleSort(T arr[], int n, int flag)
{
    int (*asc_or_dsc[])(T, T) = {less_than, greater_than};
    size_t i, j;
    for (i = 0; i < n-1; i++)
        for (j = 0; j < n-i-1; j++)
            if ((*asc_or_dsc[flag])(arr[j], arr[j+1]))
            {
                T temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
    cout << "Sorted array:\n";
    for (size_t i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;
```

```
}
```

```
// A function to check if the input character is an integer or a floating point number
```

```
int isInteger(char* str)
```

```
{
    char character;
    for(int i=0;i<strlen(str);++i)
    {
        character = str[i];
        if(character == '.')
            return 0;
    }
    return 1;
}
```

```
int main(int argc, char* argv[])
```

```
{
    // Convert the required command line arguments from string to integer
    int size = atoi(argv[1]), flag = atoi(argv[2]);
    int integer = 0;
    char a[size];
    int b[size];
    double c[size];

    if (size <= 0)
    {
        printf("Invalid size!\nPlease enter size > 0\n");
        exit(EXIT_FAILURE);
    }

    if (flag > 1)
    {
        printf("Invalid choice!\n0 - Descending, 1 - Ascending\n");
        exit(EXIT_FAILURE);
    }

    // Check if the first element of the array is an alphabet or not
    if (isalpha(argv[3][0]))
    {
        for (int i = 0; i < size; i++)
        {
```

```

        a[i] = argv[i+3][0];
    }
    bubbleSort(a, size, flag);
}

else
{
    for (int i = 0; i < size; i++)
    {
        // Check if all the elements of the array are integers.
        // Even if a single element is of float type, the array qualifies as a float array
        if (!isInteger(argv[i+3]))
        {
            integer = 1;
            break;
        }
    }

    if (integer == 0)
    {
        for (int i = 0; i < size; i++)
        {
            b[i] = atoi(argv[i+3]);
        }

        bubbleSort(b, size, flag);
    }

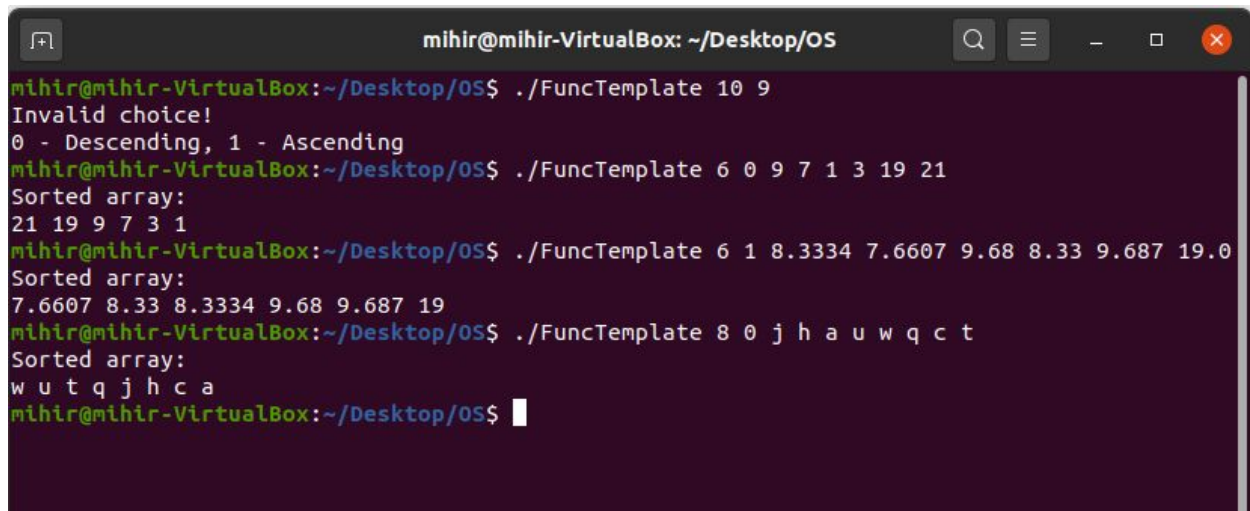
    else
    {
        for (int i = 0; i < size; i++)
        {
            c[i] = atof(argv[i+3]);
        }

        bubbleSort(c, size, flag);
    }
}

return 0;
}

```

Output screenshot:



```
mihir@mihir-VirtualBox: ~/Desktop/OS
mihir@mihir-VirtualBox:~/Desktop/OS$ ./FuncTemplate 10 9
Invalid choice!
0 - Descending, 1 - Ascending
mihir@mihir-VirtualBox:~/Desktop/OS$ ./FuncTemplate 6 0 9 7 1 3 19 21
Sorted array:
21 19 9 7 3 1
mihir@mihir-VirtualBox:~/Desktop/OS$ ./FuncTemplate 6 1 8.3334 7.6607 9.68 8.33 9.687 19.0
Sorted array:
7.6607 8.33 8.3334 9.68 9.687 19
mihir@mihir-VirtualBox:~/Desktop/OS$ ./FuncTemplate 8 0 j h a u w q c t
Sorted array:
w u t q j h c a
mihir@mihir-VirtualBox:~/Desktop/OS$
```