OPERATING SYSTEMS PRACTICE (PREPARATORY ASSIGNMENT)

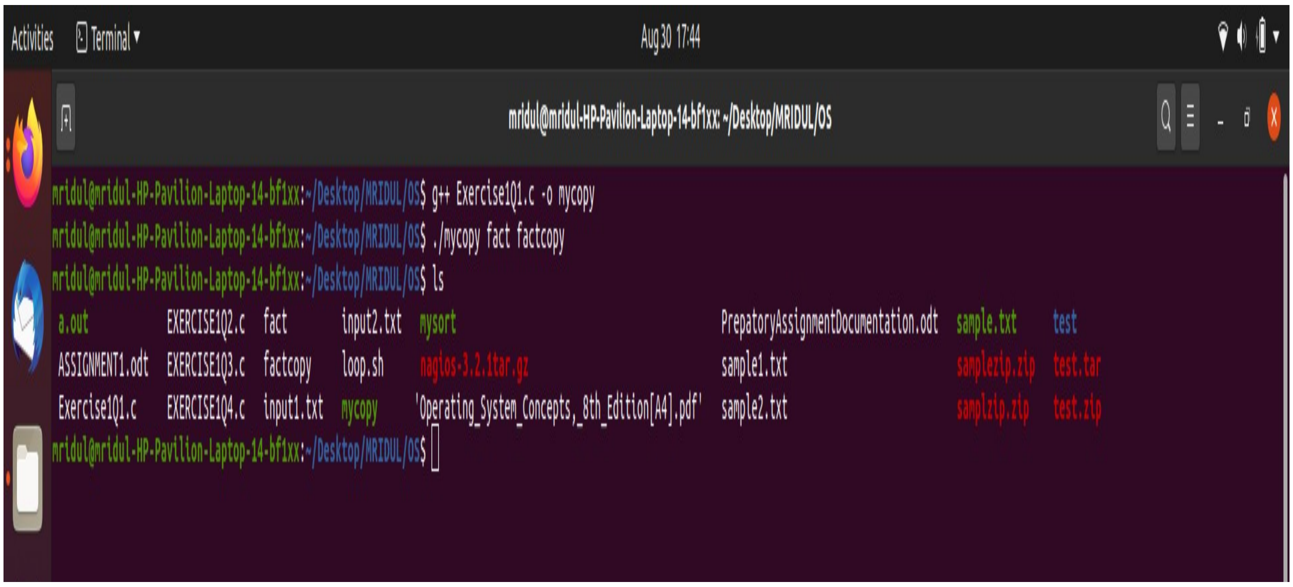
Name – Mridul Harish
Roll number – CED18I034

Q1 - 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.

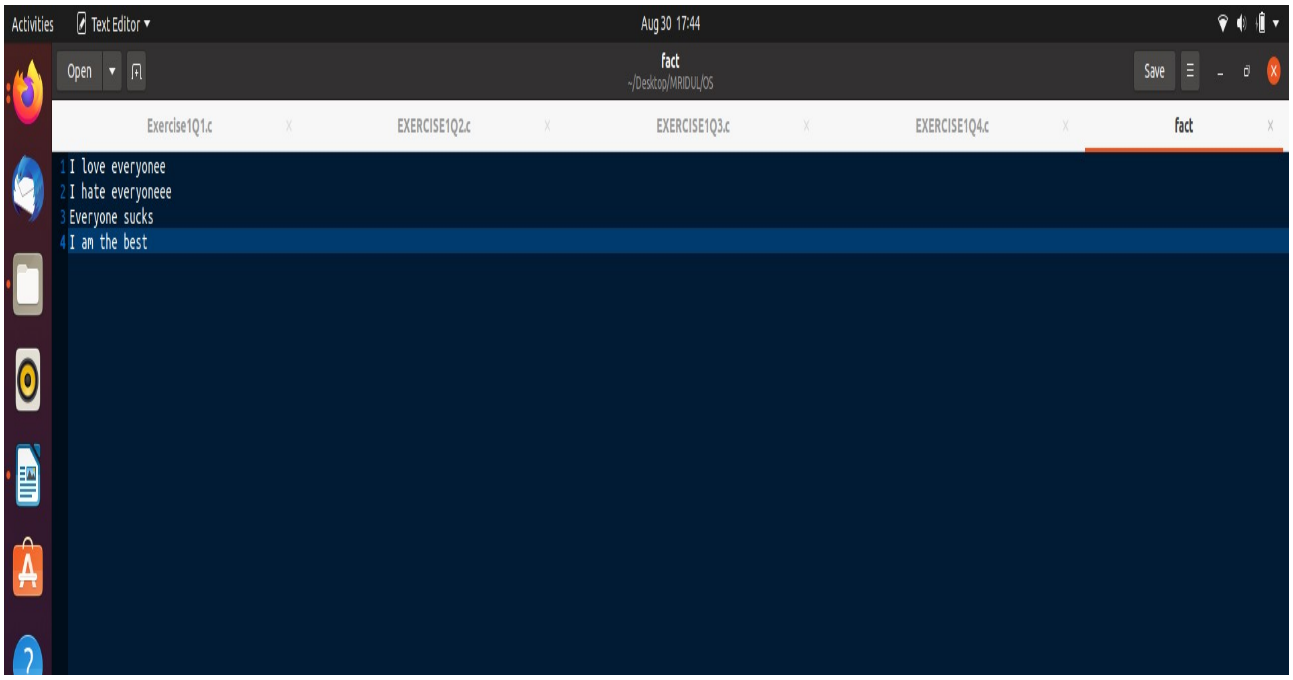
A1 – Code :-

```
#include <stdio.h>

int copy(char *, char *);
int main(int argc, char *argv[])
{
char *fn1 = argv[1]; //get the filename1 from the command line
char *fn2 = argv[2]; //get the filename2 from the command line

if (copy(fn2, fn1) == -1)
{
        printf("Error in copying\n");
        return 1;
}
   return 0;
}

int copy(char *fnDest, char *fnSrc)
{
int c;

FILE *fpDest, *fpSrc; //fpDest is the destination file pointer, fpSrc is the source file
pointer

if ((fpDest = fopen(fnDest, "w")) && (fpSrc = fopen(fnSrc, "r")))
{
        while ((c = getc(fpSrc)) != EOF)
                putc(c,fpDest);
        fclose(fpDest);
        fclose(fpSrc);

        return 0;
}
    return -1;
}
```

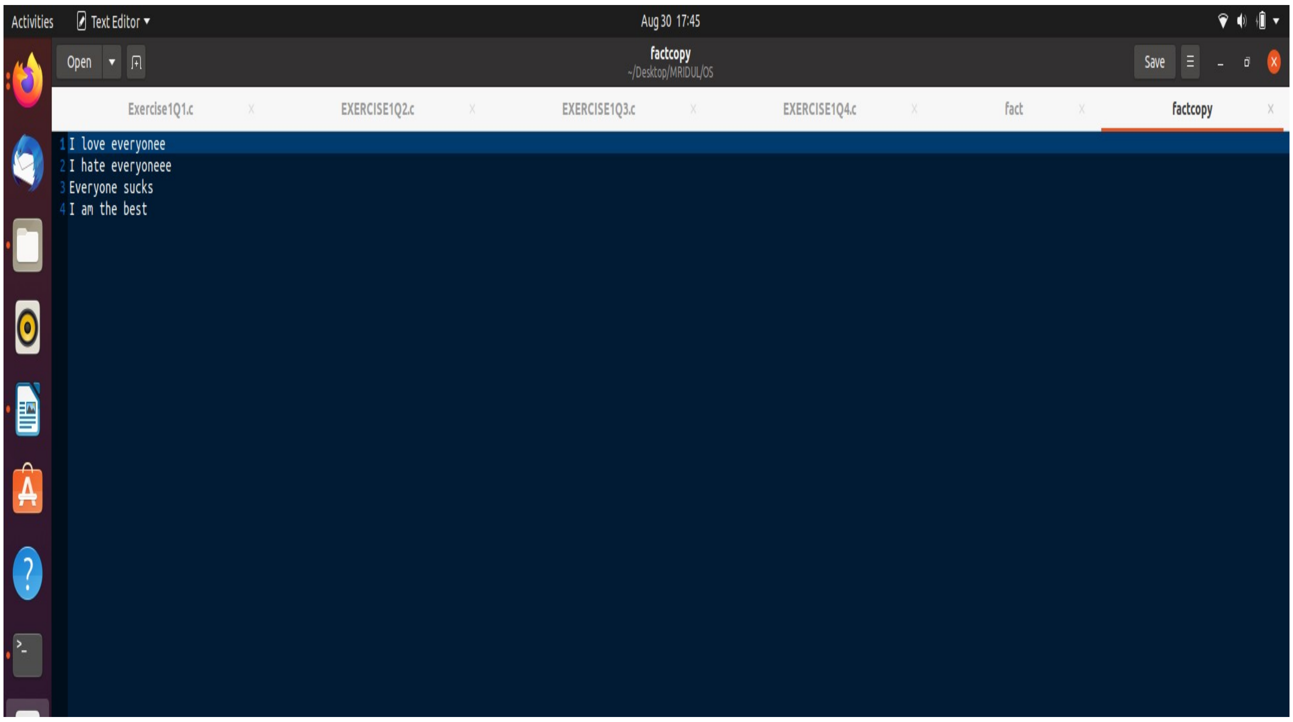
<u>Output</u> :-


Activities
Terminal
Aug 30 17:44
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx: ~/Desktop/MRIDUL/OS

```
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ g++ Exercise1Q1.c -o mycopy
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ ./mycopy fact factcopy
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ ls
 a.out           EXERCISE1Q2.c  fact      input2.txt  mysort                                        PreparatoryAssignmentDocumentation.odt  sample.txt    test
 ASSIGNMENT1.odt EXERCISE1Q3.c  factcopy  loop.sh     nagios-3.2.1tar.gz                            sample1.txt                             samplezip.zip test.tar
 Exercise1Q1.c   EXERCISE1Q4.c  input1.txt mycopy    'Operating_System_Concepts,_8th_Edition[A4].pdf' sample2.txt                           samplzip.zip  test.zip
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ 
```



```
I love everyonee
I hate everyoneee
Everyone sucks
I am the best
```



```
I love everyonee
I hate everyoneee
Everyone sucks
I am the best
```

<u>Logic</u> – In the above question we are using arguments in main. We are passing the name of both the files i.e the original file and the copy to be created in the first 2 linux arguments. We create a function copy which will the do the main process of copying the file. We use the destination and source file pointer here and open the destination file and read the source file using fopen. Then using the "getc" function, we copy the content from the source file to the destination file till we reach the End Of File(EOF). Then using fclose to close both the files. Therefore, we have simulated the functioning of the "cp" linux command.

Q2 - 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!

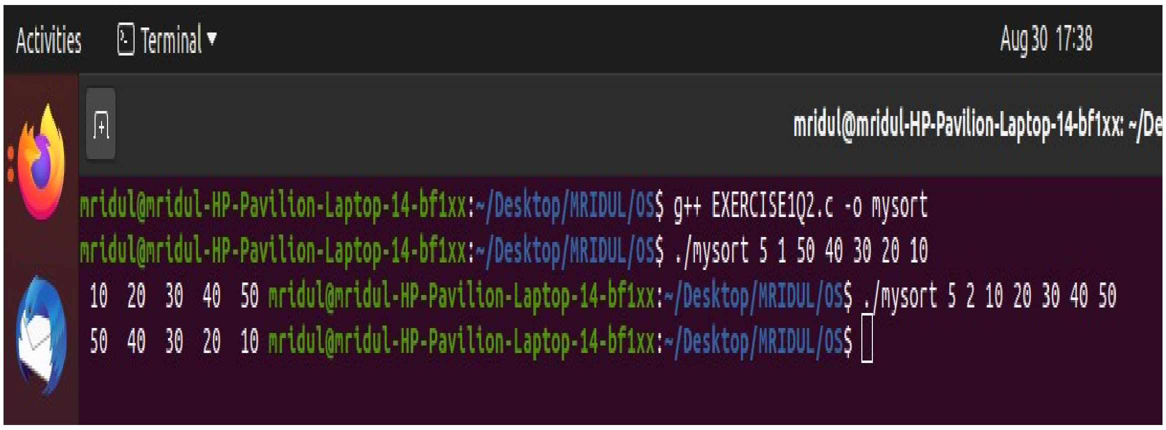
A2 – <u>Code</u> :-

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

void ascendingsort(int n, int *arr);
void descendingsort(int n, int *arr);


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

```
{
	int arr[100]; int n; int s;

	n = atoi(argv[1]);
	s = atoi(argv[2]);

	for(int i = 3; i < argc; i = i+1)
	{
		arr[i-3] = atoi(argv[i]);
	}

	if(s == 1)
		ascendingsort(n, arr);
	else if(s == 2)
		descendingsort(n, arr);
}

void ascendingsort(int n, int *arr)
{
	for (int i = 0; i < n; i = i+1)
	{
		for (int j = 0; j < n; j = j+1)
		{
			if (arr[j] > arr[i])
			{
				int tmp = arr[i];
				arr[i] = arr[j];
				arr[j] = tmp;
			}
		}
	}
	for (int i = 0; i < n; i = i+1)
		{
			printf(" %d ", arr[i]);
		}


}

void descendingsort(int n, int *arr)
{
	for (int i = 0; i < n; i = i+1)
	{
		for (int j = 0; j < n; j = j+1)
		{
```

```
                        if (arr[j] < arr[i])
                        {
                                int tmp = arr[i];
                                arr[i] = arr[j];
                                arr[j] = tmp;
                        }
                }
        }
        for (int i = 0; i < n; i =  i+1)
                {
                        printf(" %d ", arr[i]);
                }
}
```

Output :-



```
Activities   Terminal ▾                                   Aug 30 17:38
                                mridul@mridul-HP-Pavilion-Laptop-14-bf1xx: ~/De
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ g++ EXERCISE1Q2.c -o mysort
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ ./mysort 5 1 50 40 30 20 10
 10  20  30  40  50 mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ ./mysort 5 2 10 20 30 40 50
 50  40  30  20  10 mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$
```

Logic : In the above question, we make use of arguments in main. The first arguments to main is the size of the array and the second argument is the choice of ascending and descending sort. We convert the string argument to an integer argument using the “atoi” fucntion. Then using a simple if-else statement to see whether the argument chosen is ascending sort or descending sort we execute the respective functions.

- To use command line arguments, we have to give arguments to the main function in our code. Eg : int main(int argc, char **argv).
- argc refers to the number of command line arguments passed in, which includes the actual name of the program, as invoked by the user. argv contains the actual arguments, starting with index 1. Index 0 is the program name.
- The C library function FILE *fopen(const char *filename, const char *mode) opens the filename pointed to, by filename using the given mode.
- The C library function int fclose(FILE *stream) closes the stream. All buffers are flushed.
- The C library function int getc(FILE *stream) gets the next character (an unsigned char) from the specified stream and advances the position indicator for the stream.

- The C library function int putc(int char, FILE *stream) writes a character (an unsigned char) specified by the argument char to the specified stream and advances the position indicator for the stream.
- The C library function int atoi(const char *str) converts the string argument str to an integer (type int).

Q3 - 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.

A3 – <u>Code</u> :-

```
#include<iostream>
#include<stdbool.h>
#include<string.h>
#include<sstream>
#include<ctype.h>
using namespace std;

void ascendingsort(int *arr, int n);
void descendingsort(int *arr, int n);

void ascendingsort(float *arr, int n);
void descendingsort(float *arr, int n);

void ascendingsort(char *arr, int n);
void descendingsort(char *arr, int n);

bool isallLen1(int argc, char *argv[]);
bool ischar(int argc, char *argv[]);
bool isint(int argc, char *argv[]);
bool isfloat(int argc, char *argv[]);

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

        if(argc < 4)
        {
                printf("Too few arguments passed\n");
                exit(EXIT_FAILURE);
        }

        int n = atoi(argv[1]);
        int s = atoi(argv[2]);
```

```
if(s != 1 && s != 2)
{
        printf("Incorrect choice entered\n");
        exit(EXIT_FAILURE);
}

if(argc-3 != n)
{
        printf("Enter array of specified size\n");
        exit(EXIT_FAILURE);
}

int arr1[n];
float arr2[n];
char arr3[n];

if(isallLen1(argc,argv))
{
        if(ischar(argc,argv))
        {
                for(int i = 4; i <= argc; i = i+1)
                        arr3[i-4] = argv[i-1][0];
                if(s == 1)
                        ascendingsort(arr3, n);
                if(s == 2)
                        descendingsort(arr3, n);

                exit(EXIT_SUCCESS);
        }
        else
        {
                for(int i = 4; i <= argc; i = i+1)
                        arr1[i-4] = atoi(argv[i-1]);
                if(s == 1)
                        ascendingsort(arr1, n);
                if(s == 2)
                        descendingsort(arr1, n);;

                exit(EXIT_SUCCESS);
        }
}

else
{
		if(isint(argc,argv))
		{
			for(int i = 4; i <= argc; i = i+1)
				arr1[i-4] = atoi(argv[i-1]);
			if(s == 1)
				ascendingsort(arr1, n);
			if(s == 2)
				descendingsort(arr1, n);

			exit(EXIT_SUCCESS);
		}
		else if(isfloat(argc,argv))
		{
			for(int i = 4; i <= argc; i = i+1)
				arr2[i-4] = atof(argv[i-1]);
			if(s == 1)
				ascendingsort(arr2, n);
			if(s == 2)
				descendingsort(arr2, n);

			exit(EXIT_SUCCESS);
		}
		else
			{
				printf("Input is invalid\n");
				exit(EXIT_FAILURE);
			}
	}

   return 0;
}

bool isallLen1(int argc, char *argv[])
{
	for (int i = 4; i <= argc; i = i+1)
	{
		if (strlen(argv[i-1]) != 1)
			return false;
	}
	return true;
}

bool ischar(int argc, char *argv[])
{
	for(int i = 4; i <= argc; i = i+1)
```

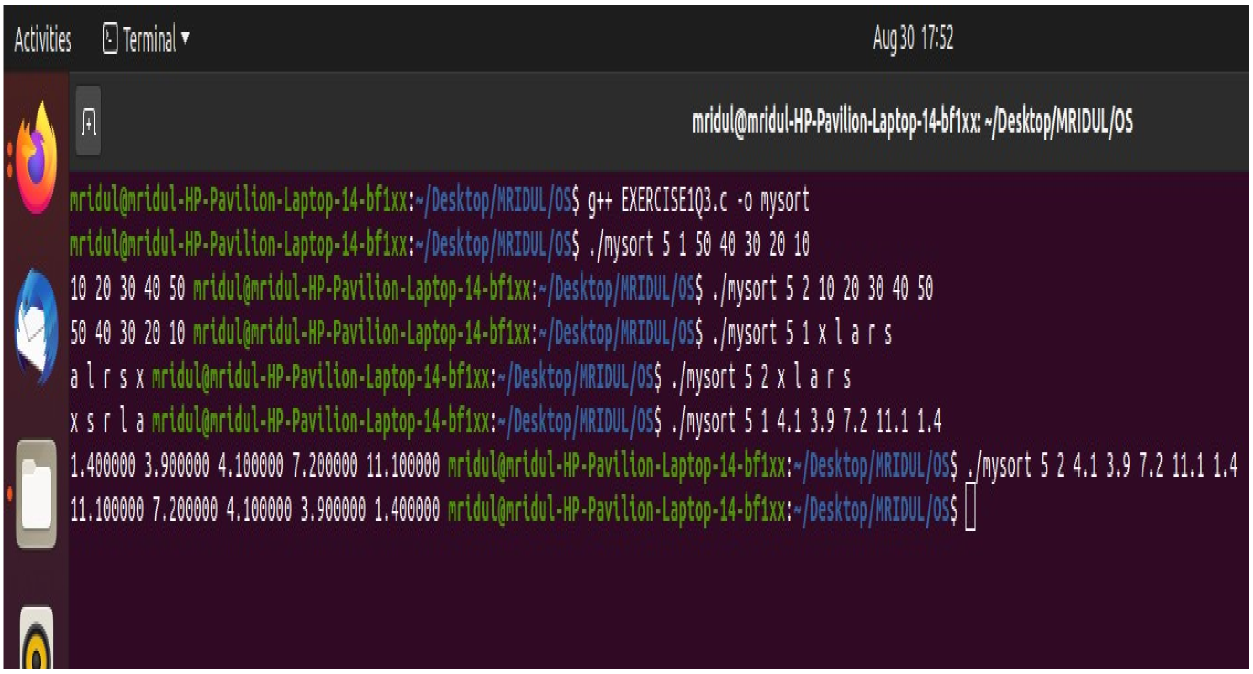
```
        {
                if(isalpha(argv[i-1][0]) != 0)
                        return true;
        }
        return false;
}

bool isint(int argc, char *argv[])
{
        int x;
        string a;
        for(int i = 4; i <= argc; i = i+1)
        {
           x = atoi(argv[i-1]);
           a = to_string(x);
           string y = string(argv[i-1]);
           if(y.compare(a) != 0)
              return false;
        }
   return true;
}

bool isfloat(int argc, char *argv[])
{
        float x;
        string a;
        for(int i = 4; i <= argc; i = i+1)
        {
                x = atof(argv[i-1]);
                stringstream ss;
                ss << x;
                a = ss.str();
                string y = string(argv[i-1]);
                if(y.compare(a) != 0)
                return false;
        }
   return true;
}

void ascendingsort(int *arr, int n)
{
        for (int i = 0; i < n; i = i+1)
        {
                for (int j = 0; j < n; j = j+1)
                {
```

```
                if (arr[j] > arr[i])
                {
                    int tmp = arr[i];
                    arr[i] = arr[j];
                    arr[j] = tmp;
                }
            }
        }
        for (int i = 0; i < n; i = i+1)
            {
                printf("%d ", arr[i]);
            }
}

void ascendingsort(char *arr, int n)
{
        for (int i = 0; i < n; i = i+1)
        {
            for (int j = 0; j < n; j = j+1)
            {
                if (arr[j] > arr[i])
                {
                    char tmp = arr[i];
                    arr[i] = arr[j];
                    arr[j] = tmp;
                }
            }
        }
        for (int i = 0; i < n; i = i+1)
            {
                printf("%c ", arr[i]);
            }
}

void ascendingsort(float *arr, int n)
{
        for (int i = 0; i < n; i = i+1)
        {
            for (int j = 0; j < n; j = j+1)
            {
                if (arr[j] > arr[i])
                {
                    float tmp = arr[i];
                    arr[i] = arr[j];
                    arr[j] = tmp;
```

```
					}
				}
			}
			for (int i = 0; i < n; i = i+1)
				{
					printf("%f ", arr[i]);
				}
}

void descendingsort(int *arr, int n)
{
			for (int i = 0; i < n; i = i+1)
			{
				for (int j = 0; j < n; j = j+1)
				{
					if (arr[j] < arr[i])
					{
						int tmp = arr[i];
						arr[i] = arr[j];
						arr[j] = tmp;
					}
				}
			}
			for (int i = 0; i < n; i =  i+1)
				{
					printf("%d ", arr[i]);
				}
}

void descendingsort(char *arr, int n)
{
			for (int i = 0; i < n; i = i+1)
			{
				for (int j = 0; j < n; j = j+1)
				{
					if (arr[j] < arr[i])
					{
						char tmp = arr[i];
						arr[i] = arr[j];
						arr[j] = tmp;
					}
				}
			}
			for (int i = 0; i < n; i =  i+1)
				{
```

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

void descendingsort(float *arr, int n)
{
        for (int i = 0; i < n; i = i+1)
        {
                for (int j = 0; j < n; j = j+1)
                {
                        if (arr[j] < arr[i])
                        {
                                float tmp = arr[i];
                                arr[i] = arr[j];
                                arr[j] = tmp;
                        }
                }
        }
        for (int i = 0; i < n; i =  i+1)
                {
                        printf("%f ", arr[i]);
                }
}
```

<u>Output</u> :


Activities
Terminal ▾
Aug 30 17:52
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx: ~/Desktop/MRIDUL/OS

```
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ g++ EXERCISE1Q3.c -o mysort
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ ./mysort 5 1 50 40 30 20 10
10 20 30 40 50 mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ ./mysort 5 2 10 20 30 40 50
50 40 30 20 10 mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ ./mysort 5 1 x l a r s
a l r s x mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ ./mysort 5 2 x l a r s
x s r l a mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ ./mysort 5 1 4.1 3.9 7.2 11.1 1.4
1.400000 3.900000 4.100000 7.200000 11.100000 mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ ./mysort 5 2 4.1 3.9 7.2 11.1 1.4
11.100000 7.200000 4.100000 3.900000 1.400000 mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$
```

<u>Logic</u> : In the above question, we implement the sort for characters and floating point numbers as well. We do this using function overloading. We have to define the ascending and descending sort functions 3-3 times each, once for int, float and char. Then we define bool functions ischar, isfloat, isint to determine whether the arguments entered are characters, intergers or floating point numbers. This we structure using a simple if-elseif-else block and then call the respective functions for ascending or descending sort.

- In the above question, we use Function Overloading.
- Function overloading is a feature in C++ where two or more functions can have the same name but different parameters.
- We also declare some fucntions using the "bool" datatype. In C++, the data type bool has been introduced to hold a boolean value, true or false.The values true or false have been added as keywords in the C++ language.
- The C library function size_t strlen(const char *str) computes the length of the string str up to, but not including the terminating null character.
- The C library function int isalpha(int c) checks if the passed character is alphabetic.
- The C++ function std::bitset::to_string() converts bitset object to string object.

Q4 - 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!

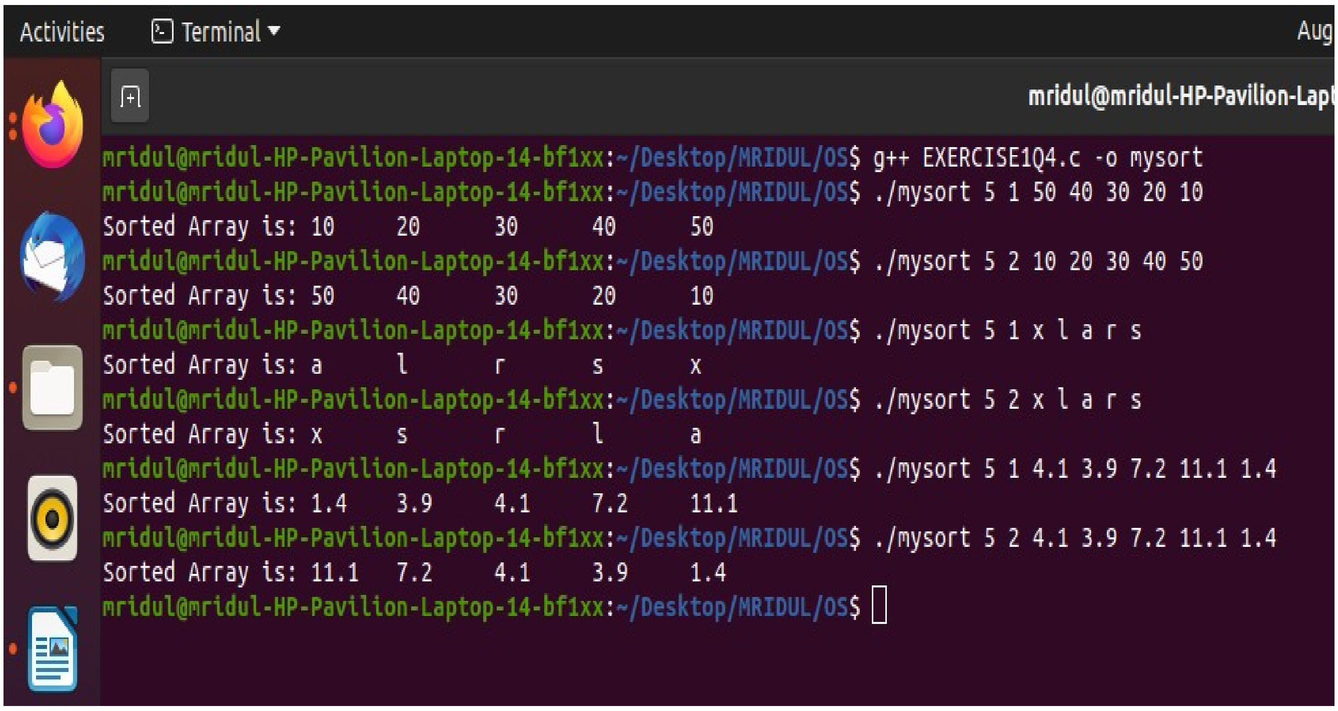
A4 – <u>Code</u> :-

```
#include<iostream>
#include<stdbool.h>
#include<string.h>
#include<sstream>
#include<ctype.h>
using namespace std;

template<typename T>
void ascendingsort(T *arr, int n)
{
        for (int i = 0; i < n; i = i+1)
        {
                for (int j = 0; j < n; j = j+1)
                {
                        if (arr[j] > arr[i])
                        {
                                T tmp = arr[i];
                                arr[i] = arr[j];
                                arr[j] = tmp;
                        }
```

```
		}
	}
}

template<typename T>
void descendingsort(T *arr, int n)
{
	for (int i = 0; i < n; i = i+1)
	{
		for (int j = 0; j < n; j = j+1)
		{
			if (arr[j] < arr[i])
			{
				T tmp = arr[i];
				arr[i] = arr[j];
				arr[j] = tmp;
			}
		}
	}
}

template<typename T>
void print(T *arr,int n)
{
	cout << "Sorted Array is: ";
	for(int i=0; i<n; i++)
	{
		cout << arr[i] << "\t";
	}
	cout << "\n";
}

bool isallLen1(int argc, char *argv[]);
bool ischar(int argc, char *argv[]);
bool isint(int argc, char *argv[]);
bool isfloat(int argc, char *argv[]);

int main(int argc, char *argv[])
{
	if(argc < 4)
	{
		printf("Too few arguments passed\n");//cout << "Few arguments passed.\
n";
		exit(EXIT_FAILURE);
	}
	int n = atoi(argv[1]);
	int s = atoi(argv[2]);

	if(s != 1 && s != 2)
	{
		printf("Incorrect choice entered\n");//cout << "Incorrect choice entered.\
n";
		exit(EXIT_FAILURE);
	}

	if(argc-3 != n)
	{
		printf("Enter array of specified size\n");//cout << "Enter array of
specified size.\n";
		exit(EXIT_FAILURE);
	}

	int arr1[n];
	float arr2[n];
	char arr3[n];

	if(isallLen1(argc,argv))
	{
		if(ischar(argc,argv))
		{
			for(int i = 4; i <= argc; i = i+1)
				arr3[i-4] = argv[i-1][0];
			if(s == 1)
				ascendingsort<char>(arr3, n);
			if(s == 2)
				descendingsort<char>(arr3, n);
			print<char>(arr3, n);
			exit(EXIT_SUCCESS);
		}
		else
		{
			for(int i = 4; i <= argc; i = i+1)
				arr1[i-4] = atoi(argv[i-1]);
			if(s == 1)
				ascendingsort<int>(arr1, n);
			if(s == 2)
				descendingsort<int>(arr1, n);
			print<int>(arr1, n);
			exit(EXIT_SUCCESS);
		}
	}

	else
	{
		if(isint(argc,argv))
		{
			for(int i = 4; i <= argc; i = i+1)
				arr1[i-4] = atoi(argv[i-1]);
			if(s == 1)
				ascendingsort<int>(arr1, n);
			if(s == 2)
				descendingsort<int>(arr1, n);
			print<int>(arr1, n);
			exit(EXIT_SUCCESS);
		}
		else if(isfloat(argc,argv))
		{
			for(int i = 4; i <= argc; i = i+1)
				arr2[i-4] = atof(argv[i-1]);
			if(s == 1)
				ascendingsort<float>(arr2, n);
			if(s == 2)
				descendingsort<float>(arr2, n);
			print<float>(arr2, n);
			exit(EXIT_SUCCESS);
		}
		else
			{
				printf("Input is invalid\n");
				exit(EXIT_FAILURE);
			}
	}

   return 0;
}

bool isallLen1(int argc, char *argv[])
{
	for (int i = 4; i <= argc; i = i+1)
	{
		if (strlen(argv[i-1]) != 1)
			return false;
	}
	return true;
```

```
}

bool ischar(int argc, char *argv[])
{
        for(int i = 4; i <= argc; i = i+1)
        {
                if(isalpha(argv[i-1][0]) != 0)
                        return true;
        }
        return false;
}

bool isint(int argc, char *argv[])
{
        int x;
        string a;
        for(int i = 4; i <= argc; i = i+1)
        {
           x = atoi(argv[i-1]);
           a = to_string(x);
           string y = string(argv[i-1]);
           if(y.compare(a) != 0)
              return false;
        }
   return true;
}

bool isfloat(int argc, char *argv[])
{
        float x;
        string a;
        for(int i = 4; i <= argc; i = i+1)
        {
                x = atof(argv[i-1]);
                stringstream ss;
                ss << x;
                a = ss.str();
                string y = string(argv[i-1]);
                if(y.compare(a) != 0)
                return false;
        }
   return true;
}
```

Output :



```
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ g++ EXERCISE1Q4.c -o mysort
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ ./mysort 5 1 50 40 30 20 10
Sorted Array is: 10     20      30      40      50
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ ./mysort 5 2 10 20 30 40 50
Sorted Array is: 50     40      30      20      10
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ ./mysort 5 1 x l a r s
Sorted Array is: a      l       r       s       x
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ ./mysort 5 2 x l a r s
Sorted Array is: x      s       r       l       a
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ ./mysort 5 1 4.1 3.9 7.2 11.1 1.4
Sorted Array is: 1.4    3.9     4.1     7.2     11.1
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ ./mysort 5 2 4.1 3.9 7.2 11.1 1.4
Sorted Array is: 11.1   7.2     4.1     3.9     1.4
mridul@mridul-HP-Pavilion-Laptop-14-bf1xx:~/Desktop/MRIDUL/OS$ 
```

Logic : In the above question, we implement the sort for integers, characters and floating point integers using function templates. The ischar, isint and isfloat functions remain the same. But instead of declaring the ascending and descending sort functions 3 times for each, we declare them only 1 time using a placeholder “T” instead of int, char and float. While the time of calling the function we just specify what is the value of T i.e. whether it is int, char or float. This helps in reducing redundancy in the code by a big amount.

- In the above question, we used function templates.
- Function templates is to pass data type as a parameter so that we don’t need to write the same code for different data types.