OPERATING SYSEMS (COM301T)

Name: Vinayak Sethi **Roll No:** COE18B061

Preparatory Programming Assignment – 1

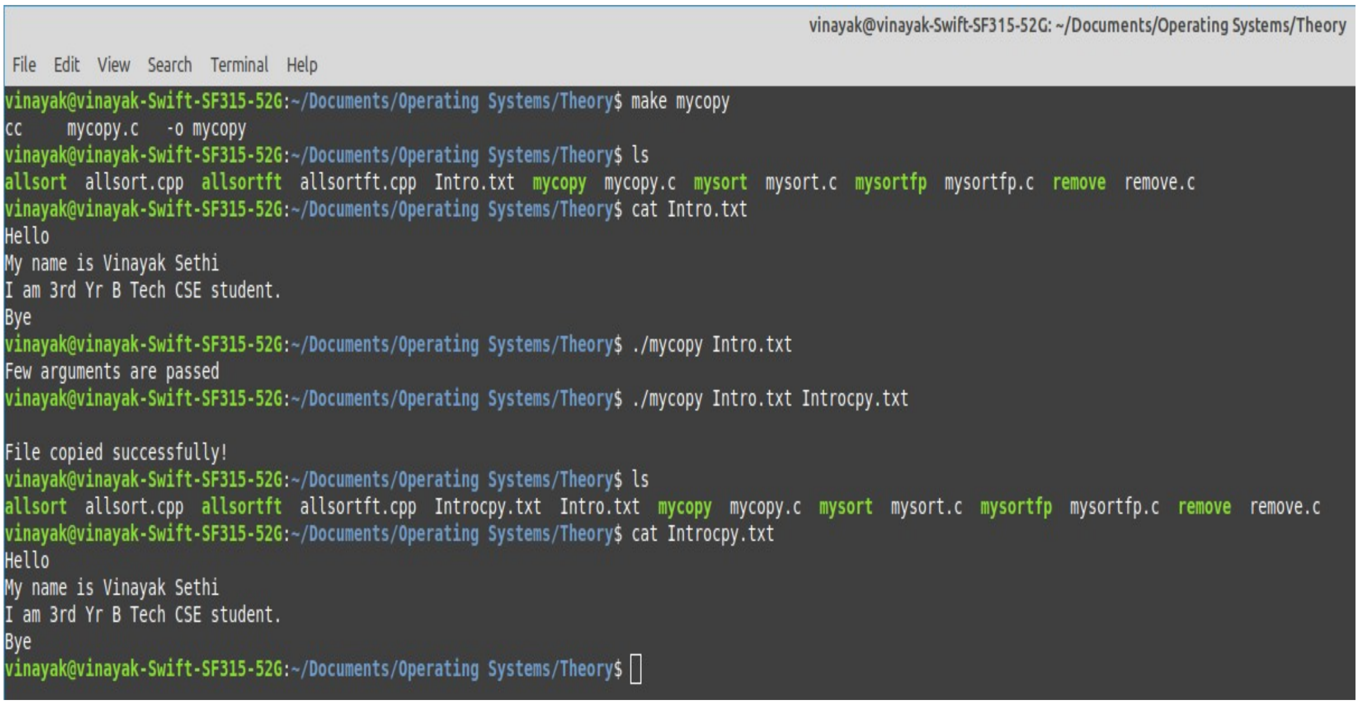
I) Develop an application (using C & Command Line Arguments) for:
i) Simulate the behavior of cp command in linux. (you should not invoke cp command from your C source!). Also your application should validate right usage; if less or more number of arguments are passed to the executable the program should prompt a message to the user. File read and write function calls are allowed. Rename your executable as **mycopy**.
Example usage could be **./mycopy fact.c factcopy.c**

Logic: In this question I have implemented **cp** command using file read and write. First it will read the content from **source file** using **fgetc** character by character until we reach **EOF**(End Of File) by opening the file in read mode and simultaneously it will write the contents character by character using **fputc** to a target file, which we give as input on command line.
Syntax: ./mycopy sourcefile targetfile
Also validation is done to check the correct usage of application using **if else** statements.

Code :- **Filename:** mycopy.c

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

int main(int argc, char ** argv)
{
	//Validate correct number of arguments
	if(argc < 3)
	{
		printf("Few arguments are passed\n");
		exit(EXIT_FAILURE);
	}
	else if(argc > 3)
	{
		printf("More arguments are passed\n");
		exit(EXIT_FAILURE);
	}
	//Copy file from One location to Another
	char ch;
	FILE *source, *target;

	source = fopen(argv[1] , "r");
	if(source == NULL)
	{
		printf("\nPress any key to exit...");
		exit(EXIT_FAILURE);
	}

	target = fopen(argv[2] , "w");
	if(target == NULL)
	{
		fclose(source);
		printf("\nError copying to target...");
		exit(EXIT_FAILURE);
	}

	while((ch = fgetc(source))!= EOF)
		fputc(ch, target);

	printf("\nFile copied successfully!\n");

	fclose(source);
	fclose(target);

	return 0;
}
```

Output:


vinayak@vinayak-Swift-SF315-52G: ~/Documents/Operating Systems/Theory
File Edit View Search Terminal Help

```
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ make mycopy
cc     mycopy.c   -o mycopy
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ls
allsort  allsort.cpp  allsortft  allsortft.cpp  Intro.txt  mycopy  mycopy.c  mysort  mysort.c  mysortfp  mysortfp.c  remove  remove.c
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ cat Intro.txt
Hello
My name is Vinayak Sethi
I am 3rd Yr B Tech CSE student.
Bye
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./mycopy Intro.txt
Few arguments are passed
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./mycopy Intro.txt Introcpy.txt

File copied successfully!
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ls
allsort  allsort.cpp  allsortft  allsortft.cpp  Introcpy.txt  Intro.txt  mycopy  mycopy.c  mysort  mysort.c  mysortfp  mysortfp.c  remove  remove.c
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ cat Introcpy.txt
Hello
My name is Vinayak Sethi
I am 3rd Yr B Tech CSE student.
Bye
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ 
```

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: In this question I have implemented **rm** command using C library function **int remove(const char *filename)** which deletes the given filename so that it is no longer accessible.

Following is the declaration for remove() function.

```
int remove(const char *filename)
```

Parameters :- filename - This is the C string containing the name of the file to be deleted.

Return Value :- On success, zero is returned. On error, -1 is returned, and errno is set appropriately.

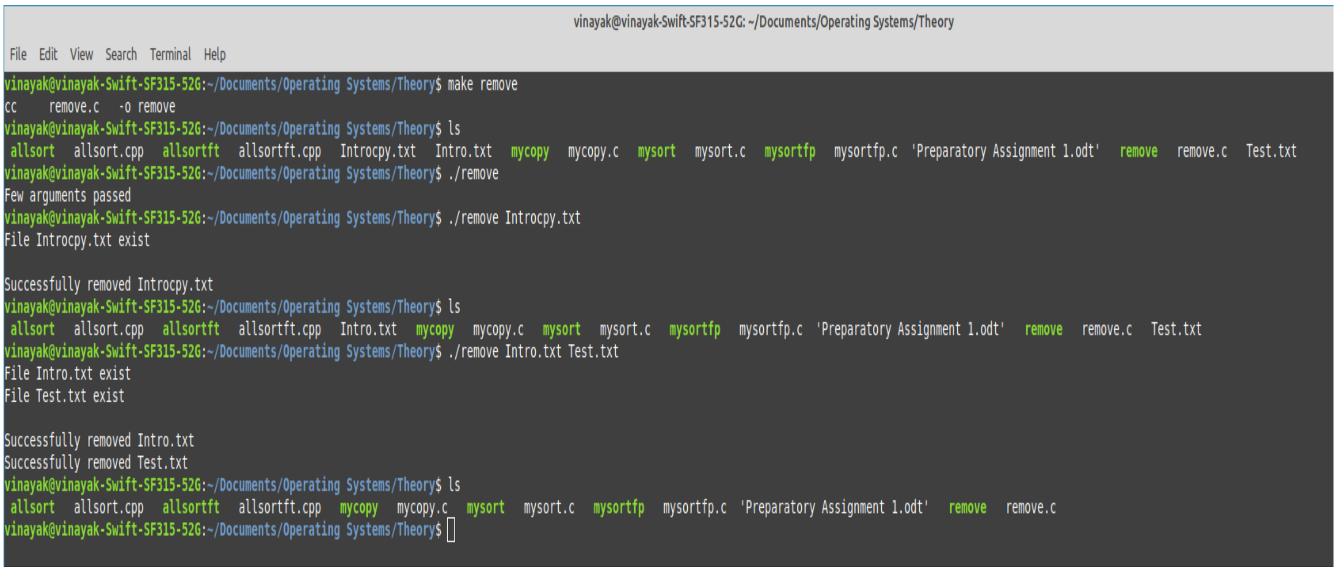
For the code implemented below
Syntax: ./remove file1name file2name
Also validation is done to check the correct usage of application using if else statements.

Code:- Filename: remove.c

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

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

        //Validate correct number of arguments
        if (argc < 2)
        {
                printf("Few arguments passed\n");
                exit(EXIT_FAILURE);
        }

        int i;

        //check if file exists or not
        for(i=1; i<=argc -1; i++)
        {
                char * filename = argv[i];
                if(access(filename,F_OK) != -1)  //check accessibilty of a file
```

```
                        printf("File %s exist\n",argv[i]);
                else
                        printf("File %s does not exist\n",argv[i]);
        }

        printf("\n");

        //remove the given files
        int del;

        for(i=1; i<= argc -1 ; i++)
        {
                del = remove(argv[i]);
                if(del == 0)
                        printf("Successfully removed %s\n",argv[i]);
                else
                        printf("Cannot remove %s\n",argv[i]);
        }

        return 0;
}
```

Output:


vinayak@vinayak-Swift-SF315-52G: ~/Documents/Operating Systems/Theory
File Edit View Search Terminal Help

```
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ make remove
cc     remove.c   -o remove
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ls
 allsort   allsort.cpp   allsortft   allsortft.cpp   Introcpy.txt   Intro.txt   mycopy   mycopy.c   mysort   mysort.c   mysortfp   mysortfp.c  'Preparatory Assignment 1.odt'   remove   remove.c   Test.txt
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./remove
Few arguments passed
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./remove Introcpy.txt
File Introcpy.txt exist

Successfully removed Introcpy.txt
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ls
 allsort   allsort.cpp   allsortft   allsortft.cpp   Intro.txt   mycopy   mycopy.c   mysort   mysort.c   mysortfp   mysortfp.c  'Preparatory Assignment 1.odt'   remove   remove.c   Test.txt
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./remove Intro.txt Test.txt
File Intro.txt exist
File Test.txt exist

Successfully removed Intro.txt
Successfully removed Test.txt
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ls
 allsort   allsort.cpp   allsortft   allsortft.cpp   mycopy   mycopy.c   mysort   mysort.c   mysortfp   mysortfp.c  'Preparatory Assignment 1.odt'   remove   remove.c
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ 
```

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 functions 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.

Logic: In this question i have implemented bubble sort for sorting where different function is implemented for ascending and descending order. The most important function used here is **atoi function** which converts a string to an integer.

The atoi function skips all white-space characters at the beginning of the string, converts the subsequent characters as part of the number, and then stops when it encounters the first character that isn't a number.

The syntax for the atoi function in the C Language is:
`int atoi(const char *nptr);`

Parameters or Arguments :- nptr - A pointer to a string to convert to an integer.
Returns :- The atoi function returns the integer representation of a string.

For the code implemented below
Syntax: ./mysort size   choice   num1 num2...
                                    (Asc/Desc)
Also validation is done to check the correct usage of application using if else statements.

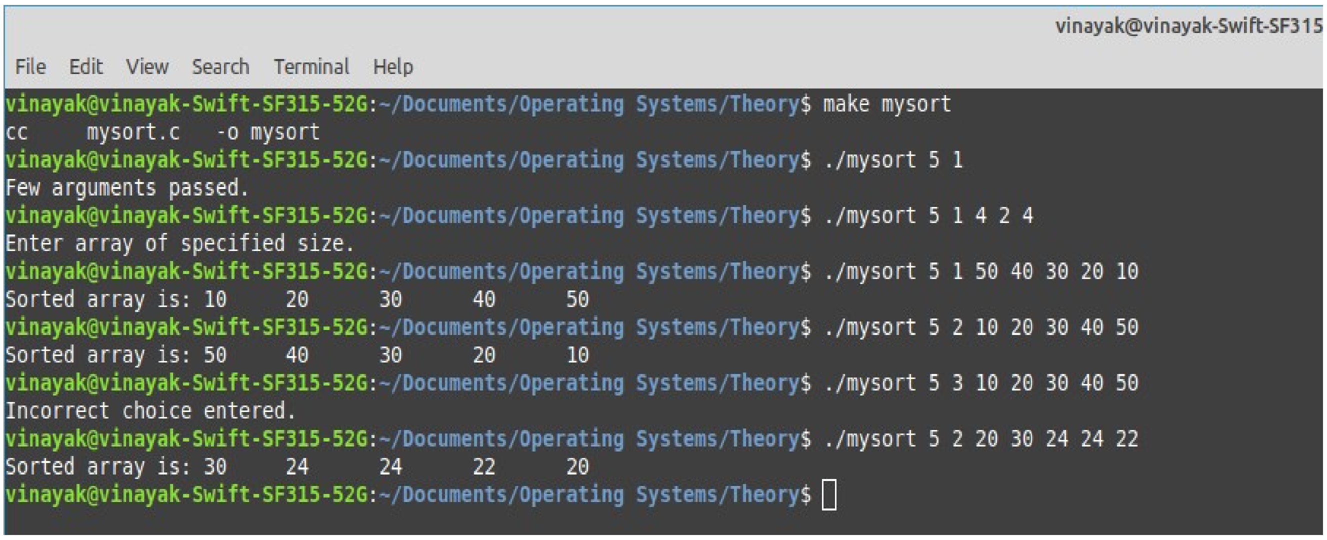
Code :- Filename: mysort.c

```
//sorting program
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

void bubblesortAsc(int *arr, int n);
void bubblesortDesc(int *arr, int n);
void swap(int *a,int *b);
void print(int *arr,int n);

int main(int argc, char ** argv)
{
        //Validate correct number of arguments
```

```
	if(argc < 4)
	{
		printf("Few arguments passed.\n");
		exit(EXIT_FAILURE);
	}

	int size = atoi(argv[1]);
	int choice = atoi(argv[2]);

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

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

	int arr[size];
	char *a;

	// convert ascii to int
	for(int i = 4; i<= argc; i++)
	{
		arr[i-4] = atoi(argv[i-1]);
		sprintf(a, "%d", arr[i-4]); //convert int to string
		if(strcmp(a,argv[i-1]) != 0)
		{
			printf("Enter only integers!\n");
			exit(EXIT_FAILURE);
		}
	}

	if(choice == 1)
		bubblesortAsc(arr,size);
	if(choice == 2)
		bubblesortDesc(arr,size);

	printf("Sorted array is: ");
	print(arr,size);
	return 0;
}

void bubblesortAsc(int *arr,int n)
{
	for(int i=0; i<n-1; i++)
	{
		for(int j=0; j<n-i-1; j++)
		{
			if(arr[j] > arr[j+1])
				swap(&arr[j],&arr[j+1]);
		}
	}
}
```

```
void bubblesortDesc(int *arr,int n)
{
        for(int i=0; i<n-1; i++)
        {
                for(int j=0; j<n-i-1; j++)
                {
                        if(arr[j] < arr[j+1])
                                swap(&arr[j],&arr[j+1]);
                }
        }
}

void swap(int *a,int *b)
{
        int c = *a;
        *a = *b;
        *b = c;
}

void print(int *arr,int n)
{
        for(int i=0; i<n; i++)
        {
                printf("%d\t",arr[i]);
        }
        printf("\n");
}
```

Output:



```
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ make mysort
cc     mysort.c    -o mysort
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./mysort 5 1
Few arguments passed.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./mysort 5 1 4 2 4
Enter array of specified size.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./mysort 5 1 50 40 30 20 10
Sorted array is: 10     20      30      40      50
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./mysort 5 2 10 20 30 40 50
Sorted array is: 50     40      30      20      10
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./mysort 5 3 10 20 30 40 50
Incorrect choice entered.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./mysort 5 2 20 30 24 24 22
Sorted array is: 30     24      24      22      20
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ 
```

Extra Credits Qn: (I would 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: This question logic is same as previous question but the main thing which is used is **function pointer** which helps us to reduce code redundancy.

Syntax to declare a function pointer

```
function_return_type(*Pointer_name)(function argument list)
```

For example: double (*p2f)(double, char)

Here double is a return type of function, p2f is name of the function pointer and (double, char) is an argument list of this function. Which means the first argument of this function is of double type and the second argument is char type.

In this question the function pointer is used to differentiate between ascending and descending order choice.

The function pointer declaration used is

```
bool(*oprn)(const void *,const void *)
```

For the code implemented below

Syntax: ./mysort size   choice   num1 num2...
                                   (Asc/Desc)

Also validation is done to check the correct usage of application using if else statements.

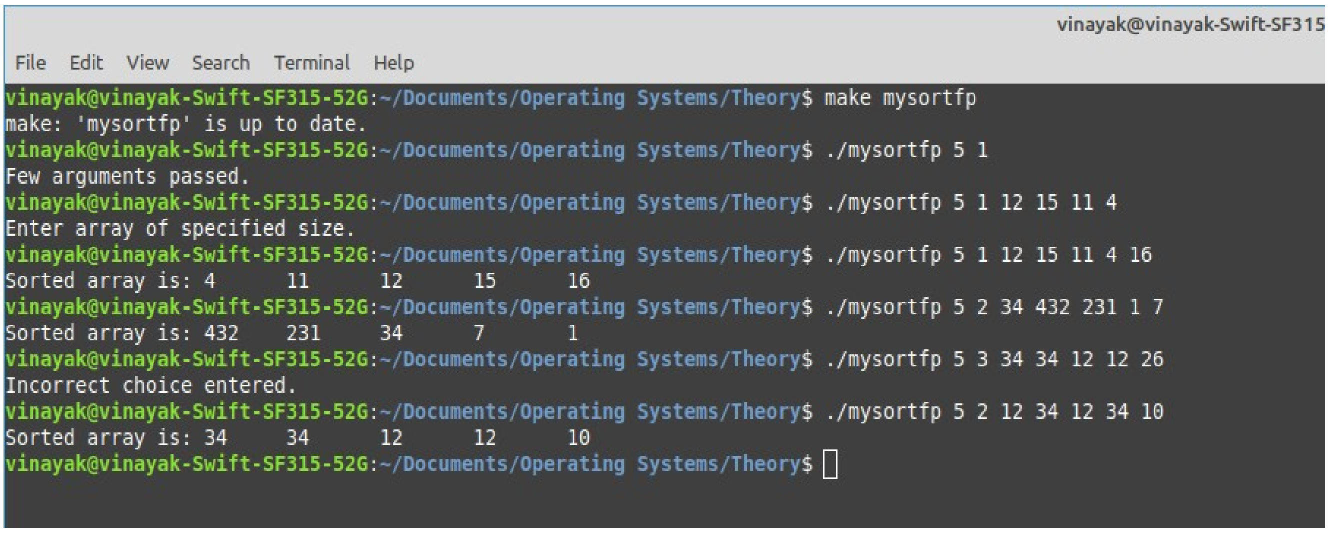
Code :- Filename: mysortfp.c

```
//sorting program using function pointer
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<stdbool.h>

void bubblesort(int *arr, int n, bool(*oprn)(const void *,const void *)); //using function pointer
bool asc(const void *a, const void *b);
bool desc(const void *a, const void *b);
void swap(int *a,int *b);
void print(int *arr,int n);
```

```
int main(int argc, char ** argv)
{
	//Validate correct number of arguments
	if(argc < 4)
	{
		printf("Few arguments passed.\n");
		exit(EXIT_FAILURE);
	}

	int size = atoi(argv[1]);
	int choice = atoi(argv[2]);

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

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

	int arr[size];
	char *a;

	// convert ascii to int
	for(int i = 4; i<= argc; i++)
	{
		arr[i-4] = atoi(argv[i-1]);
		sprintf(a, "%d", arr[i-4]); //convert int to string
		if(strcmp(a,argv[i-1]) != 0)
		{
			printf("Enter only integers!\n");
			exit(EXIT_FAILURE);
		}
	}

	if(choice == 1)
		bubblesort(arr,size,asc);
	if(choice == 2)
		bubblesort(arr,size,desc);

	printf("Sorted array is: ");
	print(arr,size);
	return 0;
}

void bubblesort(int *arr,int n, bool(*oprn)(const void *,const void *))
{
	for(int i=0; i<n-1; i++)
	{
		for(int j=0; j<n-i-1; j++)
		{
			if(oprn(&arr[j],&arr[j+1]))
```

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

bool asc(const void *a, const void *b)
{
        return *(int *)a > *(int *)b;
}

bool desc(const void *a, const void *b)
{
        return *(int *)a < *(int *)b;
}

void swap(int *a,int *b)
{
        int c = *a;
        *a = *b;
        *b = c;
}

void print(int *arr,int n)
{
        for(int i=0; i<n; i++)
        {
                printf("%d\t",arr[i]);
        }
        printf("\n");
}
```

Output:


vinayak@vinayak-Swift-SF315
File Edit View Search Terminal Help

```
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ make mysortfp
make: 'mysortfp' is up to date.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./mysortfp 5 1
Few arguments passed.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./mysortfp 5 1 12 15 11 4
Enter array of specified size.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./mysortfp 5 1 12 15 11 4 16
Sorted array is: 4      11      12      15      16
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./mysortfp 5 2 34 432 231 1 7
Sorted array is: 432    231     34      7       1
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./mysortfp 5 3 34 34 12 12 26
Incorrect choice entered.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./mysortfp 5 2 12 34 12 34 10
Sorted array is: 34     34      12      12      10
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ 
```

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.
Read on function overloading – more than one function can carry the same name with different parameters!

Logic: In this question we have used **function overloading** concept for sorting function for int, char and float data type.

Function overloading is a C++ programming feature that allows us to have more than one function having same name but different parameter list, when I say parameter list, it means the data type and sequence of the parameters.
Eg.) sum(int num1, int num2)
        sum(int num1, int num2, int num3)
        sum(int num1, double num2)

The main logic for this question is to identify the correct input data type and understand which function to call.
The **main function** contains if else block which checks if input is of length 1 or not, if it is of length 1 then it would be a character or integer of length1, else it would be a int or float.
Eg.) if input is '**a'**   **'1'**   **'b'**   **'2'**   **'c'**
then if block is implemented because all the strings are of length 1. and all will be considered character and bubble sort for characters will be implemented.
Eg.) if input is **'11.2'**   **'12'**   **'13.4'**   **'14'**
then else block is implemented and all will be considered float and bubble sort will be implemented for floating numbers input.

The important functions used is
atoi() : To convert string to integer.
atof() : To convert string to floating number.
isalpha() : TO check given character is alphabet or not.

For the code implemented below

```
Syntax: ./mysort size    choice        int_num1/     int_num2/ ...
                        (Asc/Desc)    char1/        char2/
                                      float_num1    float_num2
```

Also validation is done to check the correct usage of application using if else statements.

Code :- Filename: allsort.cpp

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

void bubblesortAsc(int *arr, int n);
void bubblesortDesc(int *arr, int n);
void swap(int *a,int *b);
void print(int *arr,int n);

void bubblesortAsc(float *arr, int n);
void bubblesortDesc(float *arr, int n);
void swap(float *a,float *b);
void print(float *arr,int n);

void bubblesortAsc(char *arr, int n);
void bubblesortDesc(char *arr, int n);
void swap(char *a,char *b);
void print(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[])
{
	//Validate correct number of arguments
	if(argc < 4)
	{
		cout << "Few arguments passed.\n";
		exit(EXIT_FAILURE);
	}

	int size = atoi(argv[1]);
	int choice = atoi(argv[2]);

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

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

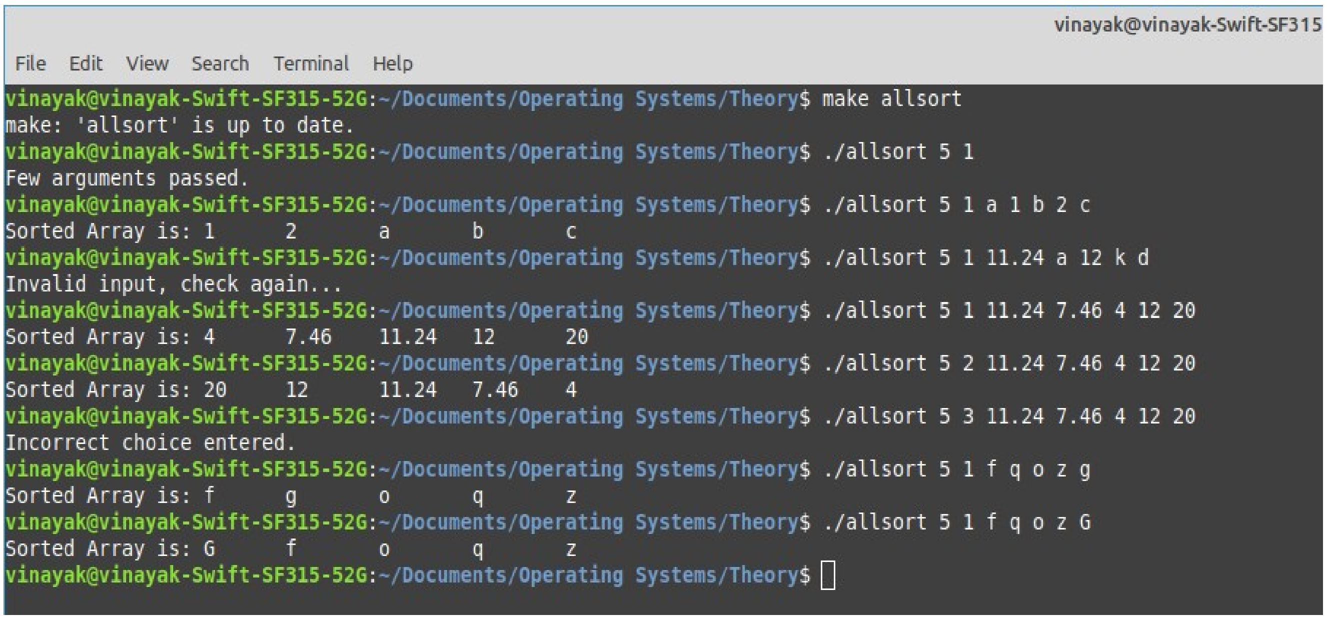
```
}

int int_arr[size];
float flt_arr[size];
char chr_arr[size];

if(isallLen1(argc,argv))
{
        if(ischar(argc,argv)) //either it is a character
        {
                for(int i=4; i<=argc; i++)
                        chr_arr[i-4] = argv[i-1][0];
                if(choice == 1)
                        bubblesortAsc(chr_arr,size);
                if(choice == 2)
                        bubblesortDesc(chr_arr,size);
                print(chr_arr,size);
                exit(EXIT_SUCCESS);
        }
        else
        {
                for(int i=4; i<=argc; i++) //or it is a integer of length 1
                        int_arr[i-4] = atoi(argv[i-1]);
                if(choice == 1)
                        bubblesortAsc(int_arr,size);
                if(choice == 2)
                        bubblesortDesc(int_arr,size);
                print(int_arr,size);
                exit(EXIT_SUCCESS);
        }
}

else
{
        if(isint(argc,argv))
        {
                for(int i=4; i<=argc; i++)
                        int_arr[i-4] = atoi(argv[i-1]);
                if(choice == 1)
                        bubblesortAsc(int_arr,size);
                if(choice == 2)
                        bubblesortDesc(int_arr,size);
                print(int_arr,size);
                exit(EXIT_SUCCESS);
        }
        else if(isfloat(argc,argv))
        {
                for(int i=4; i<=argc; i++)
                        flt_arr[i-4] = atof(argv[i-1]);
                if(choice == 1)
                        bubblesortAsc(flt_arr,size);
                if(choice == 2)
                        bubblesortDesc(flt_arr,size);
                print(flt_arr,size);
                exit(EXIT_SUCCESS);
        }
        else
```

```
                {
                        cout << "Invalid input, check again...\n";
                        exit(EXIT_FAILURE);
                }
        }

   return 0;
}

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

bool ischar(int argc, char *argv[])
{
        for(int i=4; i<=argc; i++)
        {
                if(isalpha(argv[i-1][0]) != 0) //check if it is an alphabet or not
                        return true;
        }
        return false;
}

bool isint(int argc, char *argv[])
{
        int x;
        string a;
        for(int i=4; i<=argc; i++)
        {
                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++)
        {
                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 bubblesortAsc(int *arr,int n)
{
	for(int i=0; i<n-1; i++)
	{
		for(int j=0; j<n-i-1; j++)
		{
			if(arr[j] > arr[j+1])
				swap(&arr[j],&arr[j+1]);
		}
	}
}

void bubblesortDesc(int *arr,int n)
{
	for(int i=0; i<n-1; i++)
	{
		for(int j=0; j<n-i-1; j++)
		{
			if(arr[j] < arr[j+1])
				swap(&arr[j],&arr[j+1]);
		}
	}
}

void bubblesortAsc(float *arr,int n)
{
	for(int i=0; i<n-1; i++)
	{
		for(int j=0; j<n-i-1; j++)
		{
			if(arr[j] > arr[j+1])
				swap(&arr[j],&arr[j+1]);
		}
	}
}

void bubblesortDesc(float *arr,int n)
{
	for(int i=0; i<n-1; i++)
	{
		for(int j=0; j<n-i-1; j++)
		{
			if(arr[j] < arr[j+1])
				swap(&arr[j],&arr[j+1]);
		}
	}
}

void bubblesortAsc(char *arr,int n)
{
	for(int i=0; i<n-1; i++)
	{
		for(int j=0; j<n-i-1; j++)
		{
			if(arr[j] > arr[j+1])
				swap(&arr[j],&arr[j+1]);
		}
	}
}

void bubblesortDesc(char *arr,int n)
{
	for(int i=0; i<n-1; i++)
	{
		for(int j=0; j<n-i-1; j++)
		{
			if(arr[j] < arr[j+1])
				swap(&arr[j],&arr[j+1]);
		}
	}
}

void swap(int *a,int *b)
{
	int c = *a;
	*a = *b;
	*b = c;
}

void swap(float *a,float *b)
{
	float c = *a;
	*a = *b;
	*b = c;
}

void swap(char *a,char *b)
{
	char c = *a;
	*a = *b;
	*b = c;
}

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

void print(float *arr,int n)
{
	cout << "Sorted Array is: ";
	for(int i=0; i<n; i++)
	{
		cout << arr[i] << "\t";
```

```
        }
        cout << "\n";
}

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

Output:


vinayak@vinayak-Swift-SF315
File Edit View Search Terminal Help

```
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ make allsort
make: 'allsort' is up to date.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./allsort 5 1
Few arguments passed.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./allsort 5 1 a 1 b 2 c
Sorted Array is: 1      2       a       b       c
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./allsort 5 1 11.24 a 12 k d
Invalid input, check again...
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./allsort 5 1 11.24 7.46 4 12 20
Sorted Array is: 4      7.46    11.24   12      20
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./allsort 5 2 11.24 7.46 4 12 20
Sorted Array is: 20     12      11.24   7.46    4
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./allsort 5 3 11.24 7.46 4 12 20
Incorrect choice entered.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./allsort 5 1 f q o z g
Sorted Array is: f      g       o       q       z
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./allsort 5 1 f q o z G
Sorted Array is: G      f       o       q       z
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ 
```

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: The logic used in this question is same as previous question (Q III) but the main C++ feature used is **function template**.

Function templates are special functions that can operate with *generic types*. This allows us to create a function template whose functionality can be adapted to more than one type or class without repeating the entire code for each type.

In C++ this can be achieved using ***template parameters***. A template parameter is a special kind of parameter that can be used to pass a type as argument: just like regular function parameters can be used to pass values to a function, template parameters allow to pass also types to a function. These function templates can use these parameters as if they were any other regular type.

The format for declaring function templates with type parameters is:

```
template <class identifier> function_declaration;
template <typename identifier> function_declaration;
```

We have used function templates to reduce the bubble sort function for different data types.

The important functions used is
atoi() : To convert string to integer.
atof() : To convert string to floating number.
isalpha() : TO check given character is alphabet or not.

For the code implemented below
Syntax: ./mysort size   choice   int_num1/   int_num2/ ...
                                  (Asc/Desc) char1/   char2/
                                                   float_num1   float_num2
Also validation is done to check the correct usage of application using if else statements.

Code :- Filename: allsortft.cpp

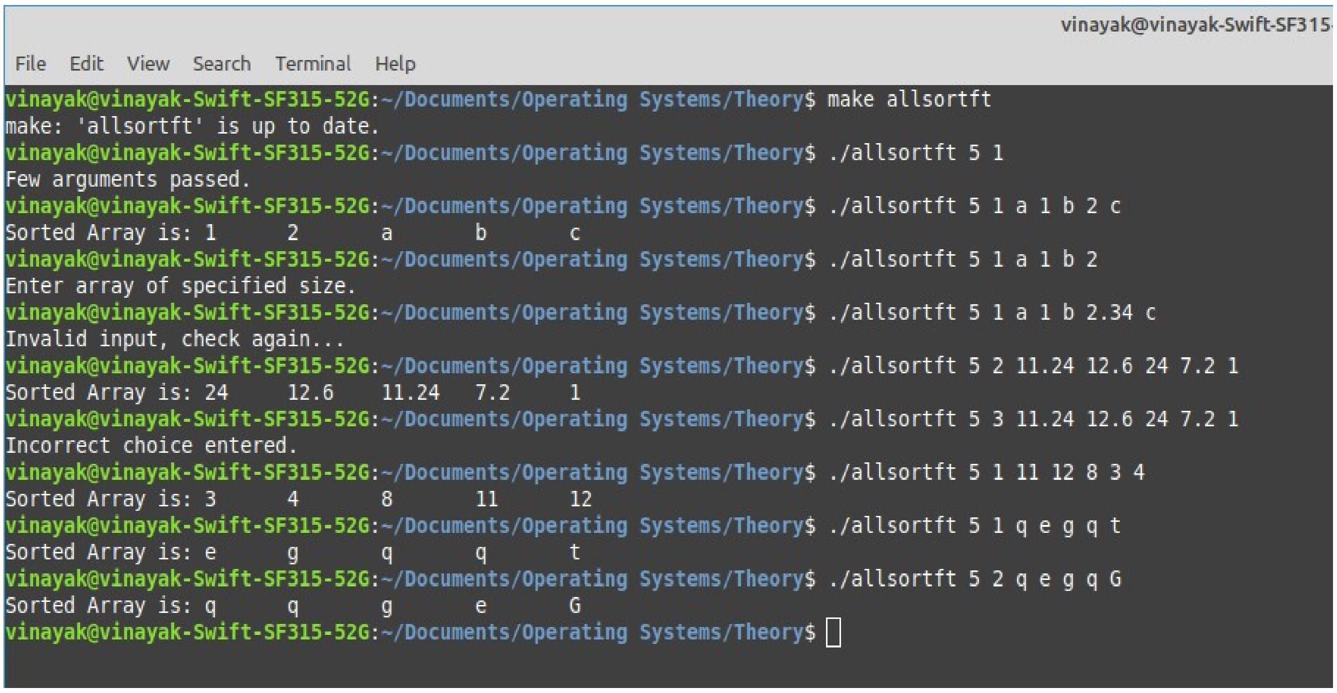
```
//sorting program using function template
#include<iostream>
#include<stdbool.h>
#include<string.h>
#include<sstream>
#include<ctype.h>
using namespace std;

template<typename T>
void bubblesortAsc(T *arr,int n) //using function template
{
        for(int i=0; i<n-1; i++)
        {
```

```
		for(int j=0; j<n-i-1; j++)
		{
			if(arr[j] > arr[j+1])
			{
				T c = arr[j+1];
				arr[j+1] = arr[j];
				arr[j] = c;
			}
		}
	}
}

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

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[])
{
	//Validate correct number of arguments
	if(argc < 4)
	{
		cout << "Few arguments passed.\n";
		exit(EXIT_FAILURE);
	}

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

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

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

int int_arr[size];
float flt_arr[size];
char chr_arr[size];

if(isallLen1(argc,argv))
{
	if(ischar(argc,argv)) //either it is a character
	{
		for(int i=4; i<=argc; i++)
			chr_arr[i-4] = argv[i-1][0];
		if(choice == 1)
			bubblesortAsc<char>(chr_arr,size);
		if(choice == 2)
			bubblesortDesc<char>(chr_arr,size);
		print<char>(chr_arr,size);
		exit(EXIT_SUCCESS);
	}
	else
	{
		for(int i=4; i<=argc; i++) //or it is a integer of length 1
			int_arr[i-4] = atoi(argv[i-1]);
		if(choice == 1)
			bubblesortAsc<int>(int_arr,size);
		if(choice == 2)
			bubblesortDesc<int>(int_arr,size);
		print<int>(int_arr,size);
		exit(EXIT_SUCCESS);
	}
}

else
{
	if(isint(argc,argv))
	{
		for(int i=4; i<=argc; i++)
			int_arr[i-4] = atoi(argv[i-1]);
		if(choice == 1)
			bubblesortAsc<int>(int_arr,size);
		if(choice == 2)
			bubblesortDesc<int>(int_arr,size);
		print<int>(int_arr,size);
		exit(EXIT_SUCCESS);
	}
	else if(isfloat(argc,argv))
	{
				for(int i=4; i<=argc; i++)
					flt_arr[i-4] = atof(argv[i-1]);
				if(choice == 1)
					bubblesortAsc<float>(flt_arr,size);
				if(choice == 2)
					bubblesortDesc<float>(flt_arr,size);
				print<float>(flt_arr,size);
				exit(EXIT_SUCCESS);
			}
			else
				{
					cout << "Invalid input, check again...\n";
					exit(EXIT_FAILURE);
				}
		}

   return 0;
}

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

bool ischar(int argc, char *argv[])
{
	for(int i=4; i<=argc; i++)
	{
		if(isalpha(argv[i-1][0]) != 0) //check if it is an alphabet or not
			return true;
	}
	return false;
}

bool isint(int argc, char *argv[])
{
	int x;
	string a;
	for(int i= 4; i<=argc; i++)
	{
		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++)
        {
                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:


vinayak@vinayak-Swift-SF315
File Edit View Search Terminal Help

```
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ make allsortft
make: 'allsortft' is up to date.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./allsortft 5 1
Few arguments passed.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./allsortft 5 1 a 1 b 2 c
Sorted Array is: 1      2       a       b       c
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./allsortft 5 1 a 1 b 2
Enter array of specified size.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./allsortft 5 1 a 1 b 2.34 c
Invalid input, check again...
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./allsortft 5 2 11.24 12.6 24 7.2 1
Sorted Array is: 24     12.6    11.24   7.2     1
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./allsortft 5 3 11.24 12.6 24 7.2 1
Incorrect choice entered.
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./allsortft 5 1 11 12 8 3 4
Sorted Array is: 3      4       8       11      12
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./allsortft 5 1 q e g q t
Sorted Array is: e      g       q       q       t
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$ ./allsortft 5 2 q e g q G
Sorted Array is: q      q       g       e       G
vinayak@vinayak-Swift-SF315-52G:~/Documents/Operating Systems/Theory$
```