Adhesh Reghu Kumar
COE18B001

Operating Systems

Preparatory Assignment - 1

Q1. i

Problem Statement

Simulate the behavior of cp command in linux using fileio. Also use validation.

Algorithm/Logic

The cp command can be emulated in C using FileIO.

The file must be opened to read and write binaries.

```
fp = fopen(src,"rb");
...
fp_w = fopen(dest,"wb");
```

Using fseek, find the length of the file(in bytes) to enable read and write.

```
fseek(fp,0,SEEK_END);
long int len = ftell(fp);
// reset the seek to the beginning.
fseek(fp,0,SEEK_SET);
```

Read the file and write to the output file given.

```
// read into the buffer. Sizeof buffer is len calc above.
fread(buffer,sizeof(buffer),1,fp);
...
// write binary to output
fwrite(buffer,sizeof(buffer),1,fp_w);
```

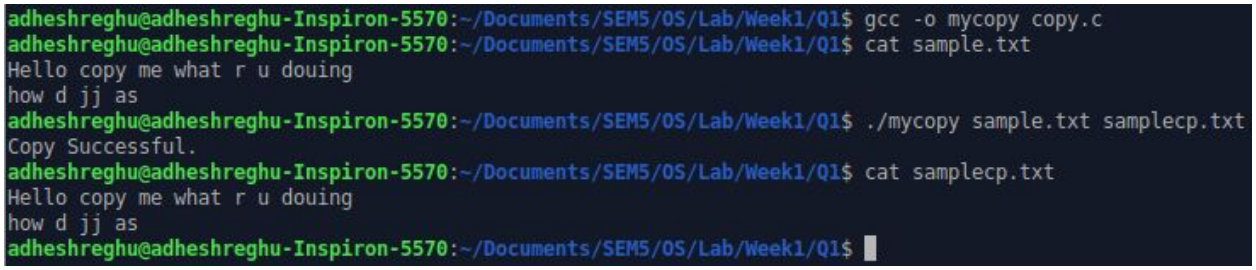
Source Code (Refer to copy.c):

```
/*****************************************************************************
* AUTHOR : AdheshR*
* Question No.: Q1.i. *
* Problem Statement: Simulate the behavior of cp command in linux.
(Just the basic version.)*
*****************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// STATUS tells the result of the copy operation.
enum STATUS
{
    SRC_INC,
    SUCCESS,
    FAILURE
};

enum STATUS fileCopy(char* src, char* dest);

int main(int argc,char* argv[])
{
    if(argc < 3)
      printf("Insufficient Arguments Passed.\n");
    else if(argc == 3)
    {
      // Get source and destination and pass to fileCopy() function.
      char* src = argv[1];
      char* dest = argv[2];
      enum STATUS status = fileCopy(src,dest);
      // Print the status of the copy.
      if(status == SRC_INC)
            printf("Source Not Found.\n");
      else if(status == SUCCESS)
            printf("Copy Successful.\n");
      else
```

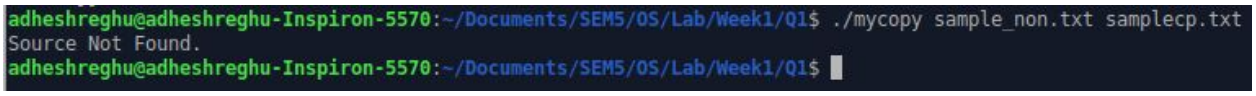
```
            printf("Error.\n");
    }
    else
       printf("Too Many Arguments.\n");
    return 0;
}

enum STATUS fileCopy(char* src, char* dest)
{
    FILE *fp;
    // Read binaries to handle any format of file.
    fp = fopen(src,"rb");

    if(fp == NULL)
       return SRC_INC;
    else
    {
       // get the size of the fp by seeking to END OF FILE
       fseek(fp,0,SEEK_END);
       long int len = ftell(fp);
       fseek(fp,0,SEEK_SET);

       // create a buffer of input size(bytes)
       unsigned char buffer[len];

       // read into buffer
       fread(buffer,sizeof(buffer),1,fp);

       // write binary to output
       FILE *fp_w;
       fp_w = fopen(dest,"wb");
       fwrite(buffer,sizeof(buffer),1,fp_w);
       fclose(fp);
       return SUCCESS;
    }
    return FAILURE;
}
```

Output:

On successful compilation where the source exists



When source does not exist



Q1. ii

Problem Statement

Simulate the behavior of rm command in linux using C. Also use validation.

Algorithm/Logic

The rm command can be emulated in C using remove function.

The file to be removed must be passed to the remove function. It returns 0 on successful deletion and 1 on failure.

```
if(remove(fileName) == 0)
{
  ++flag;
  printf("removed '%s\'\n",fileName);
}
else
  printf("cannot remove \'%s\': No such file or directory\n",fileName);
```

We can get all the filenames from argv by iterating until argc-1.

```
while(i<count)
    {
       // extract next fileName
       fileName = files[i];

       if(remove(fileName) == 0)
       {
             ++flag;
             printf("removed '%s\'\n",fileName);
       }
       else
             printf("cannot remove \'%s\': No such file or directory\n",fileName);
       i = i+1;
    }
```

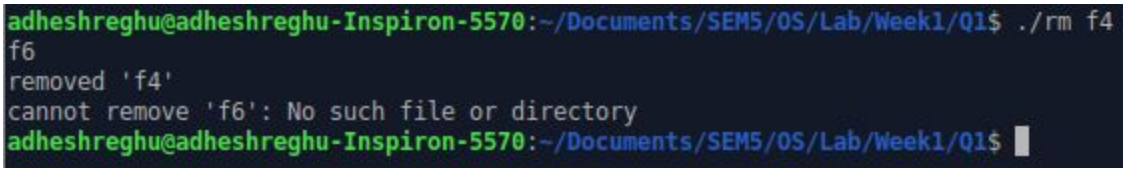
Source Code (Refer to rm.c)

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

int deleteFiles(char** files,int count);

int main(int argc, char *argv[])
{
    if(argc < 2)
      printf("Insufficient Arguments Passed.\n");
    else
    {
      int status = deleteFiles(argv,argc);
      // status is -1 if none of the files can be deleted.
      if(status == -1)
            printf("invalid arguments\n");
    }

    return 0;
}

int deleteFiles(char** files,int count)
{
    // Maintain a flag to return as status
    int flag = -1;

    // Start a loop to get all filenames in the argument passed
    int i=1;
    char *fileName;
    while(i<count)
    {
      // extract next fileName
      fileName = files[i];

      if(remove(fileName) == 0)
      {
            ++flag;
```

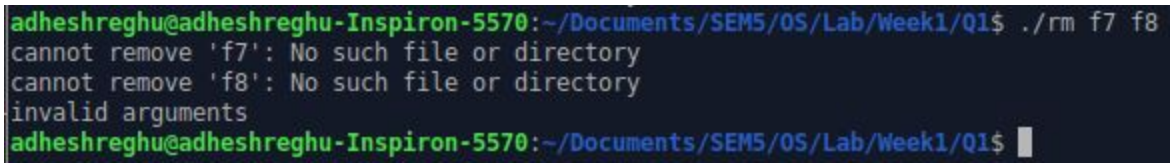
```
            printf("removed '%s\'\n",fileName);
       }
       else
            printf("cannot remove \'%s\': No such file or
directory\n",fileName);
       i = i+1;
    }
    return flag;
}
```

Output:

If valid arguments (could have some non existent files) are passed.



If only invalid arguments are passed.



Q2.

Problem Statement

Develop a program to sort an integer array. Last character decides the order.
(0 - descending, 1- ascending).

Algorithm/Logic

The sort can be achieved using a simple **bubble sort**.

The characters in argv can be converted to integers using the `atoi` function. A validation is performed before that to ensure that the character is an integer.

Validation Logic - Compare the string of characters until the end to ensure that each of the characters lie within the ASCII value for numbers.

```
// Validation Function
int isInteger(char* str)
{
    int character;
    for(int i=0;i<strlen(str);++i)
    {
       character = str[i];
       // In case of '-'' symbol, continue if not last character in
string.
       if(character == 45 && i<strlen(str)-1)
            continue;
       // Else get the decimal value (-48) and check condition.
       character -= 48;
       if(character < 0 || character > 9)
            return 0;
    }
    return 1;
}
```

The perform ascending and descending in the same sort function, the if condition is written with OR.

```
if(((order == 1) && (arr[j]>arr[j+1])) || ((order == 0) && (arr[j]<arr[j+1])))
{
.. swap
}
```

Source Code (refer to sort.c)

```
/*****************************************************
* AUTHOR : AdheshR*
* Question No.: Q2.i. *
* Problem Statement: Design the sort for integers. *
*****************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 100

void sortArray(int arr[],int n,int order);
int isInteger(char* string);

int main(int argc, char *argv[])
{
    int len = argc - 2;
    int order = atoi(argv[len+1]);
    // Ensure valid order is passed.
    if(order>1)
    {
       printf("Invalid Order Number used.\nDescending - 0, Asecending - 1.\n");
       return 0;
    }

    int arr[MAX],n=0;
    for(int i=0;i<len;++i)
    {
       // Validation function. Warning if not integer.
```

```
        if(isInteger(argv[i+1]))
                arr[n++] = atoi(argv[i+1]);
        else
                printf("warning: '%s' is not an integer. will be
ignored.\n",argv[i+1]);
    }

    if(n == 0 )
        printf("The input list is invalid. Please enter an integer array.\n");
    else
        sortArray(arr,n,order);
    return 0;
}

void sortArray(int arr[],int n,int order)
{
    int temp;
        // For swap purposes

    // Initiate Loops for simple Bubble Sort.
    for(int i=0;i<n;++i)
    {
        for(int j=0;j<n-i-1;++j)
        {
                if(((order == 1) && (arr[j]>arr[j+1])) || ((order == 0) &&
(arr[j]<arr[j+1])))
                {
                        // Swap arr[j] and arr[j+1]
                        temp = arr[j];
                        arr[j] = arr[j+1];
                        arr[j+1] = temp;
                }
        }
    }

    // Print the sorted list.
    printf("The sorted output is:\n");
    for(int i=0;i<n;++i)
        printf("%d ",arr[i]);
    puts("");
}
```

```c
// Validation Function
int isInteger(char* str)
{
    int character;
    for(int i=0;i<strlen(str);++i)
    {
       character = str[i];
       // In case of '-'' symbol, continue if not last character in string.
       if(character == 45 && i<strlen(str)-1)
             continue;
       // Else get the decimal value (-48) and check condition.
       character -= 48;
       if(character < 0 || character > 9)
             return 0;
    }
    return 1;
}
```

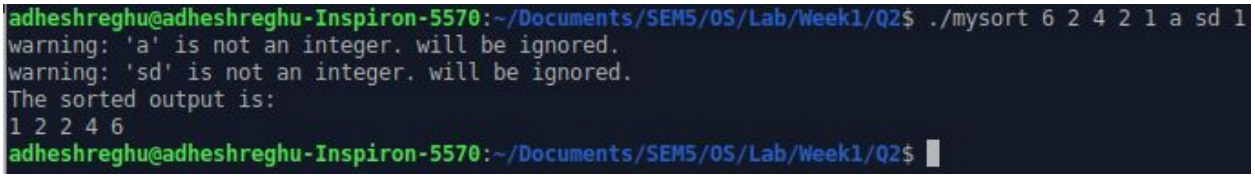
Output:

If valid characters are passed.

Descending order

Ascending Order

In case some invalid inputs are passed. (Will be ignored)



Q2. Extra Credit

Problem Statement

Develop a program to sort an integer array. Last character decides the order.
(0 - descending, 1- ascending).
Use Function Pointers.

Algorithm/Logic

The sort and validation logic is the same as Q2. The only addition to the program is the necessity to use function pointers.

The only need for a function pointer is in the compare logic. So let us define a function pointer for the same.

```
// Declare function pointer.
int (*compare[])(int,int) = {desc,asc};
```

The **asc** and **desc** are functions that take a pair of integers as parameters.
Let us take a look at them.

```
// Helper function for the compare function pointer
int desc(int a, int b)
{
    return a<b;
}

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

We use the function pointer in the **if** logic of the bubble sort algorithm.

So the modified sortArray function is

```
void sortArray(int arr[],int n,int order)
{
    // Declare function pointer.
    int (*compare[])(int,int) = {desc,asc};
    int temp;
            // For swap purposes

    // Initiate Loops for simple Bubble Sort.
    for(int i=0;i<n;++i)
    {
       for(int j=0;j<n-i-1;++j)
       {
            if((*compare[order])(arr[j],arr[j+1]))
            {
                 // Swap arr[j] and arr[j+1]
                 temp = arr[j];
                 arr[j] = arr[j+1];
                 arr[j+1] = temp;
            }
       }
    }

    // Print the sorted list.
    printf("The sorted output is:\n");
    for(int i=0;i<n;++i)
       printf("%d ",arr[i]);
    puts("");
}
```

The remaining program remains the same.

Source Code (Refer sort_fp.c):

Same as Q2 except the usage of function pointers explained above.

Output:

If valid characters are passed.

Descending order

```
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q2$ gcc -o mysort sort_fp.c -std=c99
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q2$ ./mysort 3 6 1 4 1 2 0
The sorted output is:
6 4 3 2 1 1
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q2$
```

Ascending Order

```
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q2$ ./mysort 3 6 1 4 1 2 1
The sorted output is:
1 1 2 3 4 6
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q2$
```

In case some invalid inputs are passed. (Will be ignored)

```
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q2$ ./mysort 3 6 1 4 1 2 1 a s f 0
warning: 'a' is not an integer. will be ignored.
warning: 's' is not an integer. will be ignored.
warning: 'f' is not an integer. will be ignored.
The sorted output is:
6 4 3 2 1 1 1
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q2$
```

Q3.

Problem Statement

Develop a program to sort an integer/floating point/character array.
Last character decides the order.(0 - descending, 1- ascending).
Use Function Overloading

Algorithm/Logic

The sort can be achieved using multiple **bubble sort** functions. Since integers are also part of real numbers, I have decided to build a single function to handle integers and floating points. I have handled the print for them separately.

In case the input contains int and floating point numbers the both of them are treated as floats and therefore printed as a float.

In case the input only contains int, then the array of integers is treated as such and printed as integers (not floats).

In case there is a mix of characters and int/float, the sort operation is done on both the character array and int/float array separately and the results are also printed separately.

In case of invalid entries, they are ignored and only the valid entries are considered. The differentiation is done with the help of a function `isNumber` to validate floats/int and the validation logic for characters is that anything that isn't a number and is of length 1 is considered a character.

Validation Logic - `isNumber`.

```
// Validation Function - checks for float or int.
int isNumber(char* str,int *n_i)
{
    int character,flag=0;
    for(int i=0;i<strlen(str);++i)
    {
       character = str[i];
       // In case of '-'' symbol, continue if not last character in string.
       if(character == 45 && i<strlen(str)-1)
             continue;
```

```
        // In case of '.' symbol, continue only if it occurs once.
        if(character == 46 && flag == 0)
        {
                ++flag;
                continue;
        }
        // Else get the decimal value (-48) and check condition.
        character -= 48;
        if(character < 0 || character > 9)
                return 0;
    }
    if(flag == 0)
        *n_i = *n_i+1;

    return 1;
}
```

Function Overloading is achieved by declaring and defining the same function twice but with different signatures of datatype of array passed.

```
void sortArray(float arr[],int n,int order,char rep);
void sortArray(char arr[],int n,int order);
```

char **rep** in sortArray function for float/int determines if the given array is purely int or a mixture of float and int (or purely float - both are considered as float).
It helps in the print part of the function

```
    printf("The sorted output is:\n");
    if(rep == 'i')
    {
        for(int i=0;i<n;++i)
                printf("%.0f ",arr[i]);
        puts("");
    }
    else
    {
        for(int i=0;i<n;++i)
                printf("%.2f ",arr[i]);
        puts("");
    }
```

Source Code (refer to sortOverloading.cpp)

```
#include <bits/stdc++.h>
using namespace std;
#define LL long long
#define MAX 100000
#define MOD 1000000007
#define dd double

int isNumber(char* str,int *n_i);
void sortArray(float arr[],int n,int order,char rep);
void sortArray(char arr[],int n,int order);

int main(int argc, char *argv[])
{
    int len = argc - 2;
    int order = atoi(argv[len+1]);
    // Validate order.
    if(order>1)
    {
       printf("Invalid Order Number used.\nDescending - 0, Asecending -
1.\n");
       return 0;
    }

    float arrF[MAX];
    char arrC[MAX];
    int n_f=0,n_c=0,n_i=0;

    for(int i=0;i<len;++i)
    {
       // Ensure number (int or float) or character. Else warning.
       if(isNumber(argv[i+1],&n_i))
            arrF[n_f++] = atof(argv[i+1]);
       else if((strlen(argv[i+1]) == 1))          // Any one length not a
number is a character.
            arrC[n_c++] = argv[i+1][0];
       else
```

```
            printf("warning: '%s' is not an integer/float/character. will be
ignored.\n",argv[i+1]);
    }

    // Check if purely integer array
    if((n_f - n_i)>0)
      sortArray(arrF,n_f,order,'f');
    else if(n_i>0)
      sortArray(arrF,n_i,order,'i');

    if(n_c>0)
      sortArray(arrC,n_c,order);

    if(n_c == 0 && n_f == 0)
      printf("Invalid array. The elements of the array must be
int/float/character.\n");

    return 0;
}

// Helper function for the compare funciton pointer.
int desc(float a, float b)
{
    return a<b;
}

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

void sortArray(float arr[],int n,int order,char rep)
{
    // Declare the function pointer.
    int (*compare[])(float,float) = {desc,asc};
    float temp;                                                    // for
swap purposes.
```

```
    // Initialiaze loop for the Bubble Sort.
    for(int i=0;i<n;++i)
    {
       for(int j=0;j<n-i-1;++j)
       {
            if((*compare[order])(arr[j],arr[j+1]))
            {
                 // Swap arr[j] and arr[j+1]
                 temp = arr[j];
                 arr[j] = arr[j+1];
                 arr[j+1] = temp;
            }
       }
    }

    // Print the sorted list.
    printf("The sorted output is:\n");
    if(rep == 'i')
    {
       for(int i=0;i<n;++i)
            printf("%.0f ",arr[i]);
       puts("");
    }
    else
    {
       for(int i=0;i<n;++i)
            printf("%.2f ",arr[i]);
       puts("");
    }

}

// Overload the function - for character array.
void sortArray(char arr[],int n,int order)
{
    int (*compare[])(float,float) = {desc,asc};
    char temp;
```

```
    // Bubble Sort.
    for(int i=0;i<n;++i)
    {
       for(int j=0;j<n-i-1;++j)
       {
            if((*compare[order])((int)arr[j],(int)arr[j+1]))
            {
                 // Swap arr[j] and arr[j+1]
                 temp = arr[j];
                 arr[j] = arr[j+1];
                 arr[j+1] = temp;
            }
       }
    }

    // Print the sorted list.
    printf("The sorted output is:\n");
    for(int i=0;i<n;++i)
       printf("%c ",arr[i]);
    puts("");
}


// Validation Function - checks for float or int.
int isNumber(char* str,int *n_i)
{
    int character,flag=0;
    for(int i=0;i<strlen(str);++i)
    {
       character = str[i];
       // In case of '-'' symbol, continue if not last character in string.
       if(character == 45 && i<strlen(str)-1)
            continue;
       // In case of '.' symbol, continue only if it occurs once.
       if(character == 46 && flag == 0)
       {
            ++flag;
            continue;
```

```
        }
        // Else get the decimal value (-48) and check condition.
        character -= 48;
        if(character < 0 || character > 9)
              return 0;
    }
    if(flag == 0)
        *n_i = *n_i+1;

    return 1;
}
```

It is similar to Q2.

Output:

Pure Int

```
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q3$ ./sort 1 4 2 4 8 5 7 0
The sorted output is:
8 7 5 4 4 2 1
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q3$ ./sort 1 4 2 4 8 5 7 1
The sorted output is:
1 2 4 4 5 7 8
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q3$
```

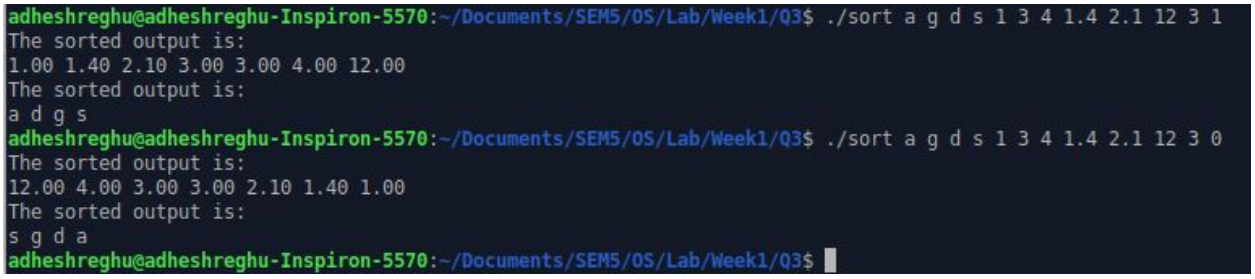
Float

```
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q3$ ./sort 1.1 -4.2 2 4.6 8 5.0 7. 1
The sorted output is:
-4.20 1.10 2.00 4.60 5.00 7.00 8.00
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q3$ ./sort 1.1 -4.2 2 4.6 8 5.0 7. 0
The sorted output is:
8.00 7.00 5.00 4.60 2.00 1.10 -4.20
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q3$
```

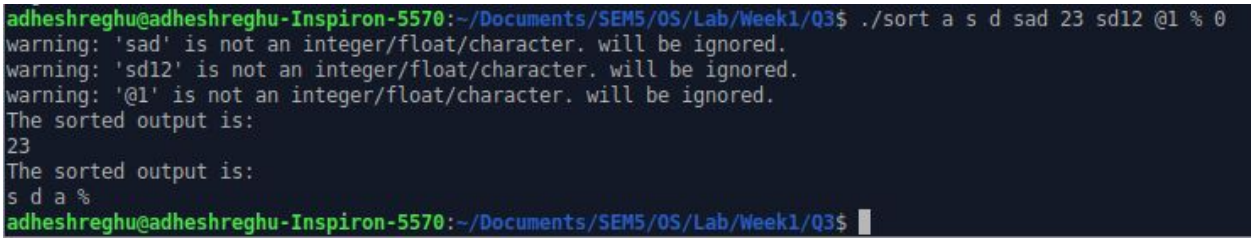
Characters

```
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q3$ ./sort m s a s c d f w 0
The sorted output is:
w s s m f d c a
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q3$ ./sort m s a s c d f w 1
The sorted output is:
a c d f m s s w
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q3$
```

Mixture



Invalid entries



Here % is also considered to be a character and is arranged in order of ASCII precedence.

Q4.

Problem Statement

Develop a program to sort an integer/floating point/character array.
Last character decides the order. (0 - descending, 1- ascending).
Use Function Templates

Algorithm/Logic

All the cases for this program are explained in Q3. The only difference is the use of **function templates** instead of overloading.

I have used two function templates namely, sortArray and swapF.
SortArray helps in sorting int/float/character arrays.
swapF is used as a part of the Bubble Sort algorithm.

Declarations of the functions are as follows.

```
// Declare function templates for sort and swap.
template<class T> void sortArray(T arr[],int n,int order);
template<class T> void swapF(T &a,T &b);
```

See the source code for the function definition.

Validation Logic is the same as used in Q3.

Source Code (Refer to sortTemplate.cpp)

```
#include <bits/stdc++.h>
using namespace std;
#define LL long long
#define MAX 100000
#define MOD 1000000007


int isNumber(char* str,int *n_i);
// Declare function templates for sort and swap.
template<class T> void sortArray(T arr[],int n,int order);
template<class T> void swapF(T &a,T &b);
```

```
int main(int argc, char *argv[])
{
    int len = argc - 2;
    int order = atoi(argv[len+1]);
    if(order>1)
    {
       printf("Invalid Order Number used.\nDescending - 0, Asecending -
1.\n");
       return 0;
    }

    float arrF[MAX];
    char arrC[MAX];
    int n_f=0,n_c=0,n_i=0;

    // Perform validation. Else warning.
    for(int i=0;i<len;++i)
    {
       if(isNumber(argv[i+1],&n_i))
             arrF[n_f++] = atof(argv[i+1]);
       else if((strlen(argv[i+1]) == 1))               // Any one length not
a number is a character.
             arrC[n_c++] = argv[i+1][0];
       else
             printf("warning: '%s' is not an integer/float/character. will
be ignored.\n",argv[i+1]);
    }

    // Check if purely integer array
    if((n_f - n_i)>0)
       sortArray<float>(arrF,n_f,order);
    else if(n_i>0)
       sortArray<float>(arrF,n_i,order);

    if(n_c>0)
       sortArray<char>(arrC,n_c,order);

    if(n_c == 0 && n_f == 0)
       printf("Invalid array. The elements of the array must be
int/float/character.\n");
    return 0;
}
```

```
template<class T> void sortArray(T arr[],int n,int order)
{
    for(int i=0;i<n;++i)
    {
       for(int j=0;j<n-i-1;++j)
       {
             if((order == 1 && (arr[j] > arr[j+1])) || (order == 0 &&
(arr[j] < arr[j+1])))
                   swapF<T>(arr[j],arr[j+1]);
       }
    }

    printf("The sorted output is:\n");
    for(int i=0;i<n;++i)
       cout << arr[i]<<" ";
    cout << endl;
}

// Validation Function
int isNumber(char* str,int *n_i)
{
    int character,flag=0;
    for(int i=0;i<strlen(str);++i)
    {
       character = str[i];
       // In case of '-'' symbol, continue if not last character in string.
       if(character == 45 && i<strlen(str)-1)
             continue;
       // In case of '.' symbol, continue only if it occurs once.
       if(character == 46 && flag == 0)
       {
             ++flag;
             continue;
       }
       // Else get the decimal value (-48) and check condition.
       character -= 48;
       if(character < 0 || character > 9)
             return 0;
    }
    if(flag == 0)
       *n_i = *n_i+1;
```

```
    return 1;
}

// Template Swap Function
template<class T> void swapF(T &a,T &b)
{
    T temp  = a;
    a = b;
    b = temp;
}
```

Output:

Pure Int

```
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q4$ g++ -o sort sortTemplate.cpp
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q4$ ./sort 1 2 4 1 6 4 3 0
The sorted output is:
6 4 4 3 2 1 1
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q4$ ./sort 1 2 4 1 6 4 3 1
The sorted output is:
1 1 2 3 4 4 6
```

Float

```
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q4$ ./sort 1 2 4. 1.1 6.6 4.908 3.44 1
The sorted output is:
1 1.1 2 3.44 4 4.908 6.6
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q4$ ./sort 1 2 4. 1.1 6.6 4.908 3.44 0
The sorted output is:
6.6 4.908 4 3.44 2 1.1 1
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q4$
```

Here since the float/int is printed using ***cout,*** therefore the integrity of both data types are preserved in the stdout without extra flag requirement.

Characters

```
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q4$ ./sort a f s q e t s 1
The sorted output is:
a e f q s s t
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q4$ ./sort a f s q e t s 0
The sorted output is:
t s s q f e a
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q4$
```

Mixture

```
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q4$ ./sort w a f d e c 9 2 5 23 4 1.45 2.42 0
The sorted output is:
23 9 5 4 2.42 2 1.45
The sorted output is:
w f e d c a
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q4$ ./sort w a f d e c 9 2 5 23 4 1.45 2.42 1
The sorted output is:
1.45 2 2.42 4 5 9 23
The sorted output is:
a c d e f w
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q4$
```

Invalid entries

```
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q4$ ./sort frw s 1 3 a 45sa 1.3 4.3 8 m 0
warning: 'frw' is not an integer/float/character. will be ignored.
warning: '45sa' is not an integer/float/character. will be ignored.
The sorted output is:
8 4.3 3 1.3 1
The sorted output is:
s m a
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q4$ ./sort frw s 1 3 a 45sa 1.3 4.3 8 m 1
warning: 'frw' is not an integer/float/character. will be ignored.
warning: '45sa' is not an integer/float/character. will be ignored.
The sorted output is:
1 1.3 3 4.3 8
The sorted output is:
a m s
adheshreghu@adheshreghu-Inspiron-5570:~/Documents/SEM5/OS/Lab/Week1/Q4$
```