Paleti Krishnasai
CED18I039

PREPARATORY PROGRAMMING ASSIGNMENT

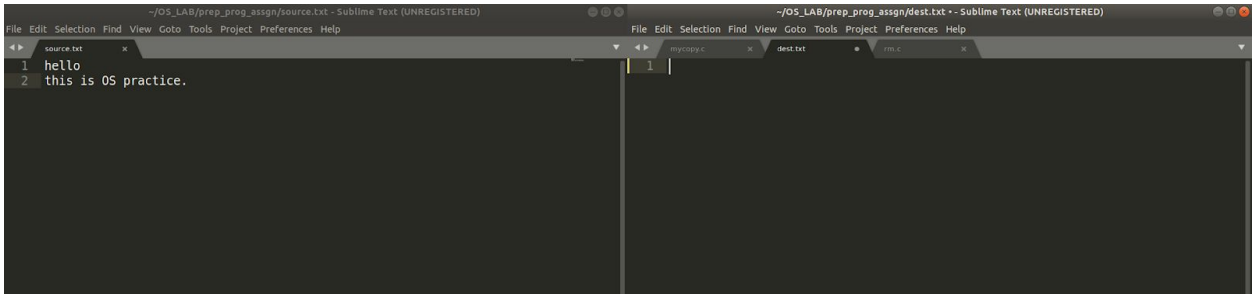
1) Simulate the behavior of cp command in linux. (you should not invoke cp command from your C source!). Also your application should validate right usage; if less or more number of arguments are passed to the executable the program should prompt a message to the user. File read and write function calls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

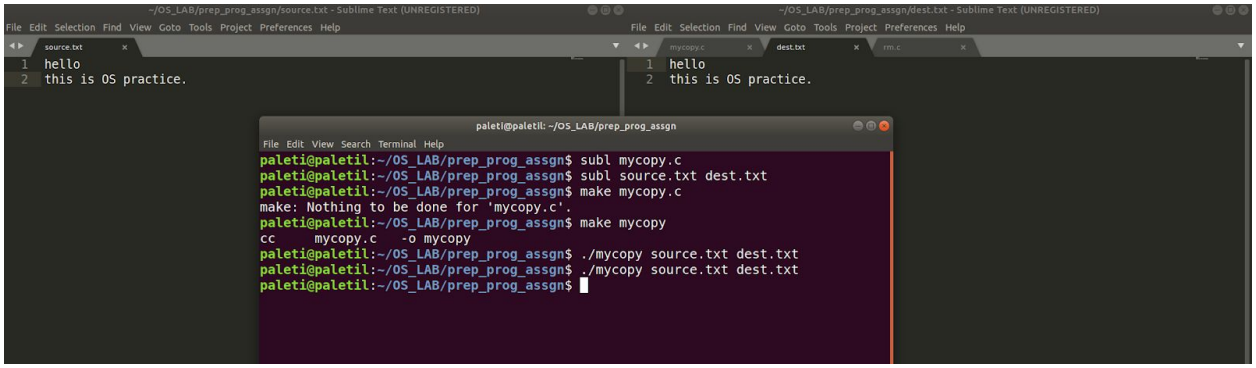
Approach :
cp command is simulated by copying characters from source file to destination file with and without using POSIX . (character by character copy), and file descriptors.

```
// Paleti Krishnasai CED18I039

// cp command using POSIX .

#include<stdio.h>
#include<stdlib.h>
#include <sys/stat.h>
#include <fcntl.h>  // file control header
#include<unistd.h> // POSIX header

int main(int argc, char *argv[])
{
    int src; //source file
      int dst; //destination file
       int n; // tracking variable
       char buf[1];

       if(argc != 3)
       {
       printf("usage ./mycopy sourceFileName destinationFileName\n");
       exit(1);
      }
      src = open(argv[1],O_RDONLY);
       if(src == -1)
       {
       perror("can't find the source file");
       exit(1);
```

Paleti Krishnasai
CED18I039

```
        }
        dst = creat(argv[2],777);
        dst = open(argv[2],O_WRONLY);
         if(dst == -1)
         {
         perror("can't create or open destination file");
         exit(1);
        }
        while((n=read(src,buf,1))>0)
        {
         write(dst,buf,1);
        }
        close(src);
        close(dst);


    /* code */
    return 0;
}
```





ALTERNATE WAY : using file pointers

Paleti Krishnasai
CED18I039

```c
// using command line arguments
// reference : Deitel & Deitel
// cp simulation WITHOUT POSIX.
#include<stdio.h>

int main(int argc, char *argv[])
{
    FILE *infileptr; // input file pointer
    FILE *outfileptr; // output file pointer
    int c; // used to hold characters read from source file.

    // cmd line arguments check
    if(argc!=3)
    {
       puts("usage : ./copy source destination");
    }
    else
    {
       if((infileptr = fopen(argv[1],"r"))!=NULL)
       {
             if((outfileptr = fopen(argv[2],"w")!=NULL))
             {
                   while((c=fgetc(infileptr))!=EOF)
                   {
                         fputc(c,outfileptr);
                   }
                   fclose(outfileptr);
             }
             else
             {
                   printf("File \"%s\" could not be opened\n",argv[2]);
             }
             fclose(infileptr);
       }
       else
       {
             printf("File \"%s\" could not be opened\n",argv[1]);
       }
   }
    /* code */
    return 0;
}
```

Paleti Krishnasai
CED18I039

Extra Credits Qn – Extend the above application / develop an application to simulate the behavior of rm command in linux. rm command invoke fromSource is not allowed! Other features as in earlier application to be supported.

Approach : same as cp , but here we are removing the file from the tree.

```c
#include<stdio.h>

void main(int argc, char* argv[]){
if(argc!=2 || argv[1]=="--help")
  {
      printf("\nusage: rm FileTodelete\n");
  }
int status;
status=remove(argv[1]);
if(status==0)
  {
      printf("successful\n");
  }
else
   {
      printf("Unsuccessful\n");
   }
}
```

```
paleti@paletil:~/OS_LAB/prep_prog_assgn$ subl myrm.c
paleti@paletil:~/OS_LAB/prep_prog_assgn$ make myrm
cc     myrm.c   -o myrm
paleti@paletil:~/OS_LAB/prep_prog_assgn$ ./myrm source.txt
successfull
paleti@paletil:~/OS_LAB/prep_prog_assgn$ ls
copy  copy.c  dest.txt  mycopy  mycopy.c  myrm  myrm.c  progprepassignment.pdf
paleti@paletil:~/OS_LAB/prep_prog_assgn$
```

2.) 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.

Paleti Krishnasai
CED18I039

Approach :
Input taken through command line arguments and bubble sort was used to simulate the sort function of linux.
The code can handle erroneous input and does both ascending and descending sort.
Atoi function is used to convert strings to integers as command line arguments are passed as characters.

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

int main(int argc, char *argv[])
{
    if(argc < 4)
    {
       printf("usage: ./mysort size order num num num...\n");
       exit(1);
    }
    int n = atoi(argv[1]);
    int order = atoi(argv[2]);
    printf("size of list : %d\n",n );
    printf("order : %d\n",order);
    if(order>2 || order<1 )
    {
       printf("usage :\n 1 -> Ascending\n2 -> Descending\n");
       exit(1);
    }
    int a[20],j=0,b[20],k=0,x,y,z;
    for (int i = 3; i <= n+2; i++)
    {
       a[j] = atoi(argv[i]);
       j++;
       /* code */
    }
    for (k= 0; k < n; k++)
    {
       b[k]=a[k];
       /* code */
    }
    if(order == 1)
```

Paleti Krishnasai
CED18I039

```
    {
       for (x=0;x<n-1;x++)
       {
             for(y=0;y<n-x-1;y++)
             {
                   if(a[y]>a[y+1])
                   {
                         z = a[y];
                         a[y] = a[y+1];
                         a[y+1] = z;
                   }
             }
       }
       printf("Sorted list in Ascending order :\n");
       for(x=0;x<n;x++)
       {
             printf("%d ",a[x]);
       }
    }
    if(order == 2)
    {
       for (x=0;x<n-1;x++)
       {
             for(y=0;y<n-x-1;y++)
             {
                   if(b[y]<b[y+1])
                   {
                         z = a[y];
                         a[y] = a[y+1];
                         a[y+1] = z;
                   }
             }
       }
       printf("Sorted list in Descending order :\n");
       for(x=0;x<n;x++)
       {
             printf("%d ",a[x]);
       }
    }

    return 0;
}
```

Paleti Krishnasai
CED18I039

```
paleti@paletil:~/OS_LAB/prep_prog_assgn$ ./mysort 3 6 10
size of list : 3
order : 6
usage :
 1 -> Ascending
2 -> Descending
```

```
paleti@paletil:~/OS_LAB/prep_prog_assgn$ make mysort
make: 'mysort' is up to date.
paleti@paletil:~/OS_LAB/prep_prog_assgn$ ./mysort 3 2 1 2 3
size of list : 3
order : 2
Sorted list in Descending order :
3 2 1 paleti@paletil:~/OS_LAB/prep_prog_assgn$ ./mysort 3 1 10 8 11
size of list : 3
order : 1
Sorted list in Ascending order :
8 10 11 paleti@paletil:~/OS_LAB/prep_prog_assgn$ ./mysort 3 1 10
```

Extra Credits Qn: (I wud advise everbody to try!)
Can you implement the above sorting (both ascending or descending) using only function internally for sorting logic (I mean bubble or insertion etc..)You should define the logic in your source code only once but the application should be able to handle both ascending or descending order sort!). Hint use function pointers!

Approach:
A simple logic of sorting a list of negative integers in ascending order will result in the positive counterparts of the integers to be in descending order.
Example : -7 -8 -9 -10 (descending order)
7 8 9 10 (ascending order)

Function Pointers are like switch case statements where in they point to a block of code rather than a value.

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
void bubblesort(int a[],int n ,int (*pointer)(int));
int negate (int a);
int retain (int a);
int main(int argc, char *argv[])
{
    if(argc < 4)
```

Paleti Krishnasai
CED18I039

```
    {
       printf("usage: ./sort size order num num num...\n");
       exit(1);
    }
    int n = atoi(argv[1]);
    int order = atoi(argv[2]);
    printf("size of list : %d\n",n );
    printf("order : %d\n",order);
    if(order>2 || order<1)
    {
       printf("usage :\n 1 -> Ascending\n2 -> Descending\n");
       exit(1);
    }
    int a[20],j=0,b[20],k=0;
    for (int i = 3; i <= n+2; i++)
    {
       a[j] = atoi(argv[i]);
       j++;
       /* code */
    }
    for (k= 0; k < n; k++)
    {
       b[k]=a[k];
    }

    int (*pointer)(int) = (order ==1)?(retain):(negate);
    bubblesort(a,n,pointer);

    for (int i = 0; i < n; i++)
    {
       printf("%d ",a[i] );
       /* code */
    }
 return 0;
}

void bubblesort(int a[],int n ,int (*pointer)(int))
{
    int x,y,z;
    for (x=0;x<n-1;x++)
       {
             for(y=0;y<n-x-1;y++)
```

Paleti Krishnasai
CED18I039

```
            {
                if(pointer(a[y])>pointer(a[y+1]))
                {
                    z = a[y];
                    a[y] = a[y+1];
                    a[y+1] = z;
                }
            }
        }
}

int negate (int a)
{
    return (-1*a);
}

int retain (int a)
{
    return (a);
}
```

```
paleti@paletil:~/OS_LAB/prep_prog_assgn$ make sortpt
make: 'sortpt' is up to date.
paleti@paletil:~/OS_LAB/prep_prog_assgn$ ./sortpt 3 2 7 3 2
size of list : 3
order : 2
7 3 2 paleti@paletil:~/OS_LAB/prep_prog_assgn$ ./sortpt 3 1 7 3 2
size of list : 3
order : 1
2 3 7 paleti@paletil:~/OS_LAB/prep_prog_assgn$ ./sortpt 3 3 7 3 2
size of list : 3
order : 3
usage :
 1 -> Ascending
2 -> Descending
paleti@paletil:~/OS_LAB/prep_prog_assgn$
```

3) 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.

Paleti Krishnasai
CED18I039

Approach :
A simple logic of sorting a list of negative integers in ascending order will result in the positive counterparts of the integers to be in descending order.
Example : -7 -8 -9 -10 (descending order)
7 8 9 10 (ascending order)

Function Pointers are like switch case statements where in they point to a block of code rather than a value.
A modification of the previous code with function overloading and function pointer to enable the same function to perform both ascending and descending sort.
Reference : https://www.geeksforgeeks.org/isalpha-isdigit-functions-c-example/ , Deitel Textbook.

```
#include<bits/stdc++.h>
using namespace std;
void bubblesort(int a[], int n, float (*pointer)(float));
void bubblesort(double a[], int n, float (*pointer)(float));
void bubblesort(char a[], int n, float (*pointer)(float));

float negate(float a);
float retain(float a);

int main(int argc, char *argv[])
{
    if (argc < 4)
    {
       printf("usage: ./sort size order num num num...\n");
       exit(1);
    }
    int n = atoi(argv[1]);
    int order = atoi(argv[2]);
    printf("size of list : %d\n", n);
    printf("order : %d\n", order);
    if (order > 2 || order < 1)
    {
       printf("usage :\n 1 -> Ascending\n2 -> Descending\n");
       exit(1);
    }

    int a[20];
    double b[20];
    char c[20];
    int check, j = 0, k = 0;
```

Paleti Krishnasai
CED18I039

```
float (*pointer)(float) = (order == 1) ? (::retain) : (::negate);

if (isdigit(argv[3][0]))
{
   check = 0;
   for (int i = 0; i < strlen(argv[3]); ++i)
   {
         if (argv[3][i] == '.')
         {
               check = 1;
               break;
         }
   }
   if (check == 0)
   {
         for (int i = 3; i <= n + 2; ++i)
         {
               a[j] = atoi(argv[i]);
               j++;
         }
         // sort
         bubblesort(a, n, pointer);
         for (int i = 0; i < n; ++i)
         {
               printf("%d ",a[i] );
               /* code */
         }
   }
   if (check == 1)
   {
         for (int i = 3; i <= n + 2; ++i)
         {
               b[j] = atof(argv[i]);
               j++;
         }
         //sort
         bubblesort(b, n, pointer);
         for (int i = 0; i < n; ++i)
         {
               printf("%lf ",b[i] );
               /* code */
         }
```

Paleti Krishnasai
CED18I039

```
        }
    }
    if (isalpha(argv[3][0]))
    {
        check = 2;
        for (int i = 3; i <= n + 2; ++i)
        {
              c[j] = argv[i][0];
              j++;
        }
        //sort
        bubblesort(c, n, pointer);
        for (int i = 0; i < n; ++i)
              {
                    printf("%c ",c[i] );
                    /* code */
              }

    }

    return 0;
}

void bubblesort(int a[], int n, float (*pointer)(float))
{
    int x, y, z;
    for (x = 0; x < n - 1; x++)
    {
        for (y = 0; y < n - x - 1; y++)
        {
              if (pointer(a[y]) > pointer(a[y + 1]))
              {
                    z = a[y];
                    a[y] = a[y + 1];
                    a[y + 1] = z;
              }
        }
    }
}
void bubblesort(double a[], int n, float (*pointer)(float))
{
    int x, y;
```

```
    double z;
    for (x = 0; x < n - 1; x++)
    {
       for (y = 0; y < n - x - 1; y++)
       {
             if (pointer(a[y]) > pointer(a[y + 1]))
             {
                   z = a[y];
                   a[y] = a[y + 1];
                   a[y + 1] = z;
             }
       }
    }
}
void bubblesort(char a[], int n, float (*pointer)(float))
{
    int x, y;
    char z;
    for (x = 0; x < n - 1; x++)
    {
       for (y = 0; y < n - x - 1; y++)
       {
             if (pointer(a[y]) > pointer(a[y + 1]))
             {
                   z = a[y];
                   a[y] = a[y + 1];
                   a[y + 1] = z;
             }
       }
    }
}
float negate(float a)
{
    return (-1 * a);
}

float retain(float a)
{
    return (a);
}
```

Paleti Krishnasai
CED18I039

```
paleti@paletil:~/OS_LAB/prep_prog_assgn$ make sortload
g++     sortload.cpp    -o sortload
paleti@paletil:~/OS_LAB/prep_prog_assgn$ make sortload
g++     sortload.cpp    -o sortload
paleti@paletil:~/OS_LAB/prep_prog_assgn$ ./sortload 3 1 3 2 1
size of list : 3
order : 1
1 2 3 paleti@paletil:~/OS_LAB/prep_prog_assgn$ ./sortload 3 1 c b a
size of list : 3
order : 1
a b c paleti@paletil:~/OS_LAB/prep_prog_assgn$ ./sortload 3 1 2.2 2.22 1.12
size of list : 3
order : 1
1.120000 2.200000 2.220000 paleti@paletil:~/OS_LAB/prep_prog_assgn$
```

4) 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 :
Same approach , but used a template.h for code generation.

```
template <class T>
T sort1(T a[],int n,float (*pointer)(float))
{
    int x,y;
    T z;
    for ( x = 0; x < n-1; ++x)
    {
       for (y = 0; y < n-x-1; ++y)
       {
             if (pointer(a[y])>pointer(a[y+1]))
             {
                   z = a[y];
                   a[y] = a[y+1];
                   a[y+1] = z;
             }
       }
    }
}
```

Paleti Krishnasai
CED18I039

```
#include<bits/stdc++.h>
using namespace std;
#include "template.h"
float negate(float a);
float retain(float a);

int main(int argc, char *argv[])
{
    if (argc < 4)
    {
       printf("usage: ./sort size order num num num...\n");
       exit(1);
    }
    int n = atoi(argv[1]);
    int order = atoi(argv[2]);
    printf("size of list : %d\n", n);
    printf("order : %d\n", order);
    if (order > 2 || order < 1)
    {
       printf("usage :\n 1 -> Ascending\n2 -> Descending\n");
       exit(1);
    }

    int a[n];
    double b[n];
    char c[n];
    int check, j = 0, k = 0;
    float (*pointer)(float) = (order == 1) ? (::retain) : (::negate);

    if (isdigit(argv[3][0]))
    {
       check = 0;
       for (int i = 0; i < strlen(argv[3]); ++i)
       {
             if (argv[3][i] == '.')
             {
                   check = 1;
                   break;
             }
       }
       if (check == 0)
       {
```

```
            for (int i = 3; i <= n + 2; ++i)
            {
                  a[j] = atoi(argv[i]);
                  j++;
            }
            // sort
            sort1(a, n, pointer);
            for (int i = 0; i < n; ++i)
            {
                  printf("%d ",a[i] );
                  /* code */
            }
      }
      if (check == 1)
      {
            for (int i = 3; i <= n + 2; ++i)
            {
                  b[j] = atof(argv[i]);
                  j++;
            }
            //sort
            sort1(b, n, pointer);
            for (int i = 0; i < n; ++i)
            {
                  printf("%lf ",b[i] );
                  /* code */
            }
      }
   }
   if (isalpha(argv[3][0]))
   {
      check = 2;
      for (int i = 3; i <= n + 2; ++i)
      {
            c[j] = argv[i][0];
            j++;
      }
      //sort
      sort1(c, n, pointer);
      for (int i = 0; i < n; ++i)
            {
                  printf("%c ",c[i] );
```

Paleti Krishnasai
CED18I039

```
                    /* code */
                }

    }
    return 0;
}

float negate(float a)
{
    return (-1 * a);
}

float retain(float a)
{
    return (a);
}
```

```
paleti@paletil:~/OS_LAB/prep_prog_assgn$ ./sort 3 2 1 2 5
size of list : 3
order : 2
5 2 1 paleti@paletil:~/OS_LAB/prep_prog_assgn$ ./sort 3 2 a b c
size of list : 3
order : 2
c b a paleti@paletil:~/OS_LAB/prep_prog_assgn$ ./sort 3 2 1.2 9.5 22.035
size of list : 3
order : 2
22.035000 9.500000 1.200000 paleti@paletil:~/OS_LAB/prep_prog_assgn$ ./sort 3 1 1.2 9.5 22.035
size of list : 3
order : 1
1.200000 9.500000 22.035000 paleti@paletil:~/OS_LAB/prep_prog_assgn$ ./sort 3 1 a b c
size of list : 3
order : 1
a b c paleti@paletil:~/OS_LAB/prep_prog_assgn$ ./sort 3 1 6 2 5
size of list : 3
order : 1
2 5 6 paleti@paletil:~/OS_LAB/prep_prog_assgn$
```