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

// FUNCTION TO THROW ERRORS => INFLUENCED BY GRAPHQL QUERY ERROR REPORTING
void throwError(char *message, char *file)
{
    if (file)
        fprintf(stderr, "%s [%s]: Error code %i: %s\n", message, file, errno, strerror(errno));
    else
        printf("%s. Proper usage of the arguments is: [-b ###] [-o outfile]\n", message);
    exit(-1);
}

int main(int argc, char *argv[])
{
    int buffer = 4096, flag, fdo, fdi, amtRead, amtWritten = 0;
    char *outfile;
    // FIND THE [-b ###] AND [-o OUTFILE] OPTIONS USING GETOPT
    while ((flag = getopt(argc, argv, "b:o:")) != -1)
        switch (flag)
        {
        case 'b':
            buffer = atoi(optarg);
            break;
        case 'o':
            outfile = optarg;
            break;
        case '?':
            throwError("Error: Unknown argument supplied", NULL);
        default:
            throwError("Error: The arguments '-b' and '-o' require arguments", NULL);
        }
    char *buff = malloc((sizeof(char)) * buffer);
    //OPEN THE OUTFILE IF ONE IS SPECIFIED
    if (outfile)
    {
        fdo = open(outfile, O_WRONLY | O_CREAT | O_TRUNC, 0666);
        if (fdo < 0)
            throwError("Error: Unable to open the output file", outfile);
    }
    else
        fdo = STDOUT_FILENO;
```

```
48      // CHECK IF ZERO INPUT => IF TRUE, TREAT IT LIKE A "-" BY APPENDING "-" TO
    ARGV
49      if (optind == argc)
50      {
51          argc = 0, optind = 0;
52          argv[argc++] = "-";
53      }
54      //ITERATE THROUGH ARGUMENTS STARTING AT CURRENT OPTIND+1 ENDING AT LAST
    ARGUMENT
55      for (; optind < argc; ++optind)
56      {
57          if (!strcmp("-", argv[optind]))
58          {
59              fdi = STDIN_FILENO;
60              argv[optind] = "stdin";
61          }
62          // OPEN INPUT FILE FOR READ
63          else if ((fdi = open(argv[optind], O_RDONLY, 0666)) < 0)
64          {
65              throwError("Error: Unable to open the input file", argv[optind]);
66          }
67          // THE READ AND WRITE OPERATIONS WITH CORRECTION FOR PARTIAL WRITES
68          while ((amtRead = read(fdi, buff, (sizeof(char)) * buffer)) != 0)
69          {
70              if (amtRead < 0)
71              {
72                  throwError("Error: Could not read from the input file",
    argv[optind]);
73              }
74              else
75              {
76
77                  while (amtWritten < amtRead)
78                  {
79                      if ((amtWritten = write(fdo, buff, amtRead)) < 0)
80                      {
81                          throwError("Error: Could not write to the output
    file", outfile);
82                      }
83
84                      amtRead = amtRead - amtWritten;
85                      buff = buff + amtRead;
86                      amtWritten = 0;
87                  }
88              }
89          }
90      }
91      // CLOSE THE OUTPUT FILE IF IT ISNT STD OUT
92      if (fdo != STDOUT_FILENO && close(fdo) < 0)
```

```
93          throwError("Error: Could not close the output file", outfile);
94      return 0;
95  }
96
```