```c
//Joe Sloyan, Robert Campbell, Aaron Sala

#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <stdio.h>

void writeOut(int fd_out, unsigned char count, int block_size, char* Block){
    //Writes a block with the correct count out to a file. One byte count then the bloc
k
    write(fd_out, &count, 1);
    write(fd_out, Block, block_size);
}

void decompress(int fd_in, int fd_out, int compression_size){
    //decompress the previously compressed file
    unsigned char count;
    char buffer[compression_size];
    while(1){
        read(fd_in, &count, 1); //read number of repetitions
        int n = read(fd_in, buffer, compression_size);  //read the number of characters
 for the repeated pattern
        if(n == 0){
            break;
        }
        for(int i = 0; i < count; i++){
            write(fd_out, buffer, n);   //write the read string n times to out file
        }
    }
    return ;
}

void compress(int fd_in, int fd_out, int compression_size){
    char buffer[compression_size];
    char lastRead[compression_size];
    unsigned char count = 1;
    int last_size = read(fd_in, lastRead, compression_size);
    //Takes care of the case if there is less than one COMPRESSION_SIZE total of data r
ead in
        if(last_size < compression_size)
        {
            write(fd_out, &count, 1);
            write(fd_out,lastRead,last_size);
            return;
        }
    while(1)
    {
        //get new set of characters to compare
        int current_size = read(fd_in, buffer, compression_size);
        if(current_size == 0){
            writeOut(fd_out, count, compression_size, lastRead);  //END OF FILE
            return;
        }
        else if(current_size < compression_size)
        {
            writeOut(fd_out, count, compression_size, lastRead);  //NOT A FULL BLOCK
            count = 1;
            write(fd_out, &count, 1);
            write(fd_out,buffer,current_size);
            return;
```

```
        }
        if(strncmp(buffer, lastRead, compression_size) == 0 && count<255){  //if theyre
 equal and the count isnt maxed out
           count++;
        }else{
            writeOut(fd_out, count, compression_size, lastRead);
            memcpy(lastRead, buffer, compression_size);  //write the buffer into lastRe
ad
            count = 1;                                          //count reset to 1
        }
    }
}


int main(int argc, char *argv[]){
    if(argc != 5){
        char errorMessage[] = "ERROR: Wrong Number Of Arguments\nUSAGE: rle <input file
> <output file> <compression length> <mode>\n\tinput file: the file to compress/decompr
ess\n\toutput file: the result of the operation\n\tcompression length: the base size of
 candidate runs\n\tmode: specifies whether to compress or decompress- if mode=0, then c
ompress the input file, if mode=1 then decompress the input file\n";
        perror(errorMessage);   //Check wrong number of arguments
        return -1;
    }
    int compress_size = atoi(argv[3]);  //get the compress size and mode
    int mode = atoi(argv[4]);
    if((access(argv[1], R_OK)) !=0){
        char errorMessage[] = "ERROR: Ivalid File\nFile Does Not Exist or user does not
 have Read Permission.";
        perror(errorMessage);   //check for permission errors on the file
        return -1;
    }
    int fd_in = open(argv[1], O_RDONLY);
    int fd_out = open(argv[2], O_WRONLY | O_CREAT | O_TRUNC , S_IRWXU); //open the file
 in writeover mode and create if it doesnt exist
    if(compress_size < 1){
        char errorMessage[] = "ERROR: Ivalid Compression Size\nCompression size must be
 at least 1.";   //check compression size
        perror(errorMessage);
        return -1;
    }



    if(mode == 0){
        compress(fd_in, fd_out, compress_size);
    }else if(mode == 1){
        decompress(fd_in,fd_out,compress_size);
    }else{
        char errorMessage[] = "ERROR: Ivalid Mode Argument\nUSAGE: rle <input file> <ou
tput file> <compression length> <mode>\n\tinput file: the file to compress/decompress\n
\toutput file: the result of the operation\n\tcompression length: the base size of cand
idate runs\n\tmode: specifies whether to compress or decompress- if mode=0, then compre
ss the input file, if mode=1 then decompress the input file\n";
        perror(errorMessage);
        return -1;
    }

    close(fd_in);
    close(fd_out);
}
```