Nidhi Patel and Malik Aneer
Assignment 2
Readme

Processes
First we give the program a file (text.txt). We take the test.txt and we take the parameters given. Then we complete error checks to make sure that enough parts are being given. Using fopen we open the text file and find the number of parts in the textfile. Then, we create buffers to copy strings and we get partition sizes. Then, we have buffers to convert the arguments into char* for exec. Exec needs a path and for our code to work the path must be ."/compressRworker." We need to store the process id's in an array and then we fork and create children. Each child receives an exec command and is given parameters: filename, buff, partbuff, startbuff, psizebuff. Then, the worker processes take over. In this program we call LOLS in the main. In the main method we have the parameters and we are converting them into integers. Then we create the new files with the compressed parts. We need to rename these files because they will contain the thread output. Then, we use the LOLs compression. We make sure that we are compressing properly when it comes to one and two repetitions. We use fprintf to print to the file. Finally, we close the file.

Threads
For the threads program, we used parent processes and child processes along with three different files. This program compresses the given string pointed to and returns a pointer to the new string. First we have a struct for the thread arguments (Char* filename Char*input int part int start and int size) We initialize a struct and unlike processes we do not make a child file. Each thread only does one function. We give it a pointer to the LOLs function and then since we cannot give it multiple arguments we initialize a struct that will have all the arguments and later we will remove all these arguments. We have a condition wait which is key. The struct is changing before the threads can store the values. After each thread is created we give is a wait condition, and as soon as everything in the struct tis stored in the threads function. We send a signal to the wait condition to unlock. We then join all the threads together.