```
#include "mmult.h"

#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <ctype.h>

// Program to perform matrix multiply and print results
// Writes to a txt file the input matrices and output matrix
// Prints time it takes to do matrix multiplication
// Usage: ./mmult (rows in matrix 1) (columns in matrix 1) (rows in matrix 2) (columns in m
atrix 2)
int main(int argc, char *argv[])
{
    uint32_t i;
    uint32_t j;
    uint32_t row1;
    uint32_t col1;
    uint32_t row2;
    uint32_t col2;

    // Txt file names
    char m1string[10] = "m1";
    char m2string[10] = "m2";
    char m3string[10] = "m3";

    // Input and output matrices
    struct matrix *m1;
    struct matrix *m2;
    struct matrix *mresult;

    // Used for timing
    struct timespec start;
    struct timespec finish;
    FILE *f;
    double time;

    // Error Checking for the right # of arguments, valid arguments, valid numbers, and tha
t the matrix multiply can be done
    // atoi can't detect if a number contains a decimal point or other characters if it sta
rts with a number. Will still work though.
    if (argc != 5 || atoi(argv[1]) <= 0 || atoi(argv[2]) <= 0 || atoi(argv[3]) <= 0 || atoi
(argv[4]) <= 0) {
        printf("Usage: ./mmult (rows in matrix 1) (columns in matrix 1) (rows in matrix 2)
(columns in matrix 2)\n");
        return -1;
    }

    for (i = 1; i < 5; i++) {
        for (j = 0; j < strlen(argv[i]); j++) {
            if (isdigit(argv[i][j]) == 0) {
                printf("Please enter a valid number!\n");
                return -2;
            }
        }
    }

    if (atoi(argv[2]) != atoi(argv[3])) {
        printf("Columns for matrix 1 must match rows for matrix 2.\n");
        return -3;
    }

    // Convert strings to int and generates input matrices with dimensions
    row1 = atoi(argv[1]);
    col1 = atoi(argv[2]);
    row2 = atoi(argv[3]);
    col2 = atoi(argv[4]);
```

```
    m1 = mgen(row1,col1);
    m2 = mgen(row2,col2);

    // Prints input matrices
    mprint(m1);
    mprint(m2);

    // Times matrix multiply
    clock_gettime(CLOCK_REALTIME, &start);
    mresult = mmult(m1,m2);
    clock_gettime(CLOCK_REALTIME, &finish);

    // Prints the time it took to do the matrix multiplication and result matrix
    mprint(mresult);
    time = (double)(finish.tv_sec - start.tv_sec) + (double)(finish.tv_nsec - start.tv_nsec
)/(double)1000000000;
    printf("Time taken for mmult (sec): %lf\n", time);

    // Creates txt files to store matrices in and time
    createtxt(m1,m1string);
    createtxt(m2,m2string);
    createtxt(mresult,m3string);

    // Write time to a text file
    f = fopen("time.txt","w+");
    fprintf(f,"%lf",time);
    fclose(f);
    printf("Created time.txt\n");

    // Frees allocated memory
    free(m1->matrix);
    free(m1);
    free(m2->matrix);
    free(m2);
    free(mresult->matrix);
    free(mresult);

    return 0;
}
```