1)

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

int main()

{

FILE \*inputA; //file stream for first input

FILE \*inputB; //file stream for second input

FILE\* output = fopen("outC.txt", "w"); //file stream for output file

char \*inputFileA = "inA.txt";

char \*inputFileB = "inB.txt";

if((inputA = fopen(inputFileA, "r")) == NULL) //open the file and check if it fails

{

fprintf(output, "File %s failed to open\n", inputFileA); //output error to file

fclose(output);

return 1; //return 1, program fail

}

if((inputB = fopen(inputFileB, "r")) == NULL) //open the file and check if it fails

{

fprintf(output, "File %s failed to open\n", inputFileB);

fclose(inputA);

fclose(output);

return 1; //return 1, program fail

}

int rowA, rowB, colA, colB;

fscanf(inputA, "%d", &rowA); //get first value from the file (rowA)

fscanf(inputA, "%d", &colA); //get the second value from the file (colA)

fscanf(inputB, "%d", &rowB); //get first value from the file (rowB)

fscanf(inputB, "%d", &colB); //get second value from the file (colB)

if(colA != rowB) //if dimensions for multiplication do not matchup

{

fprintf(output, "ERROR: dimesnsions of matricies do not match\n"); //output error to file

fclose(inputA);

fclose(inputB);

fclose(output);

return 1;

}

else

{

float\*\* matrixA; //create input matrixA

matrixA = malloc(rowA\*sizeof(float\*)); //allocate space for pointers in each row

for(int i=0;i<rowA;i++)

matrixA[i] = malloc(colA\*sizeof(matrixA[0])); //allocate space for columns in the 2D array

float\*\* matrixB; //create input matrixB

matrixB = malloc(rowB\*sizeof(float\*));

for(int i=0;i<rowB;i++)

matrixB[i] = malloc(colB\*sizeof(matrixB[0]));

float\*\* matrixOut; //create output matrixOut

matrixOut = malloc(rowA\*sizeof(float\*));

for(int i=0;i<rowA;i++)

matrixOut[i] = malloc(colB\*sizeof(matrixOut[0]));

float temp; //temp variable for buffer for scanning input files

for(int r=0;r<rowA;r++)

{

for(int c=0;c<colA;c++)

{

fscanf(inputA, "%f", &temp); //look for next float in the file

matrixA[r][c] = temp; //load it into the matrixA

}

}

for(int r=0;r<rowB;r++)

{

for(int c=0;c<colB;c++)

{

fscanf(inputB, "%f", &temp); //look for floats in file

matrixB[r][c] = temp; //load into matrixB

}

}

for(int r=0;r<rowA;r++)

{

for(int c=0;c<colB;c++)

{

for(int z=0;z<rowB;z++) //compute RnCnA\*RnCnB + Rn+1Cn+1A\*Rn+1Cn+1B z is for acessing rows in colB

{

matrixOut[r][c] = matrixOut[r][c] + (matrixA[r][z] \* matrixB[z][c]);

}

}

}

fprintf(output, "%d %d\n", rowA, colB); //write dimensions for matrixOut to outfile

for(int r=0;r<rowA;r++)

{

for(int c=0;c<colB;c++)

{

fprintf(output, "%f ", matrixOut[r][c]); //write matrixOut to outfile

}

}

}

fclose(inputA); //close all files

fclose(inputB);

fclose(output);

return 0;

}

Output for A:

2 2

20.160000 27.779999 44.309998 56.039997

Output for B:

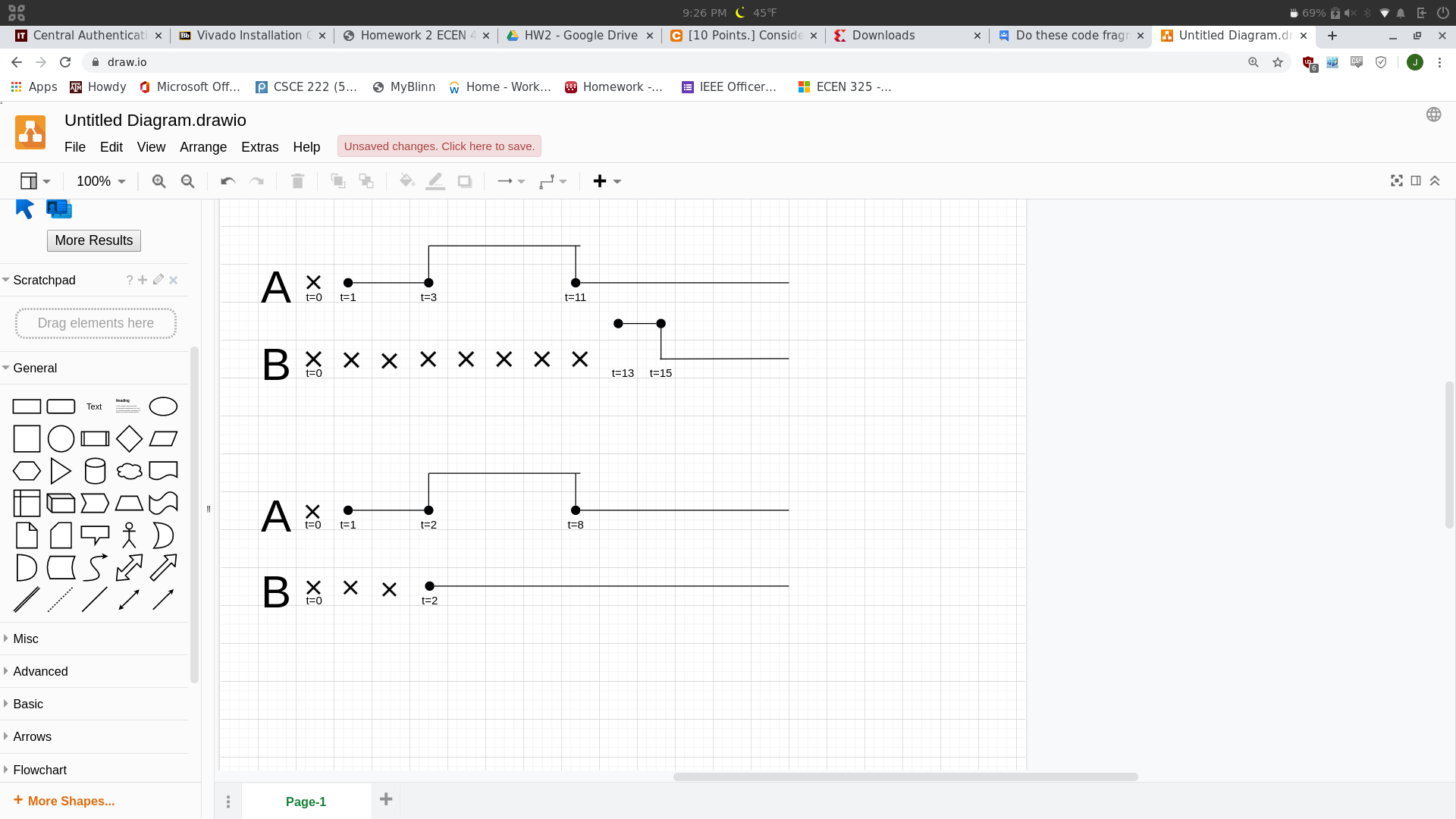
ERROR: dimesnsions of matricies do not match

2)

1. Final (executable), main.o, block1.o, block2.o
2. If only block1.c is modified, when make is recalled, only block1.c is recompiled, thus block1.o is regenerated. It does not have to regenerate all of the files because they have not been modified, so they are still correctly linked in the final. Once block1.o is regenerated, it will be linked again inside of final with all of the other files. Since a relinking is happening, the final (executable) file is regenerated.
3. Since the file main.c is already created, the touch command will only edit the timestamp of the file. The makefile will detect this new change and recompile the file. This will cause main.o to be regenerated. The final (executable) file will also be regenerated, because a relinking must happen since main.o has been regenerated.

3)

Fragment A



Fragment B

