Lab 1 – CPE 333

Group members & Contributions:

Carlie Vargas: Worked on the logic and structuring of the code, also made sure that calling conventions were being followed. Verified codes correctness alongside Karina and Maddie

Karina Lee: Helped with formatting and getting the proof of our working code and worked with Carlie on the starter code and layout.

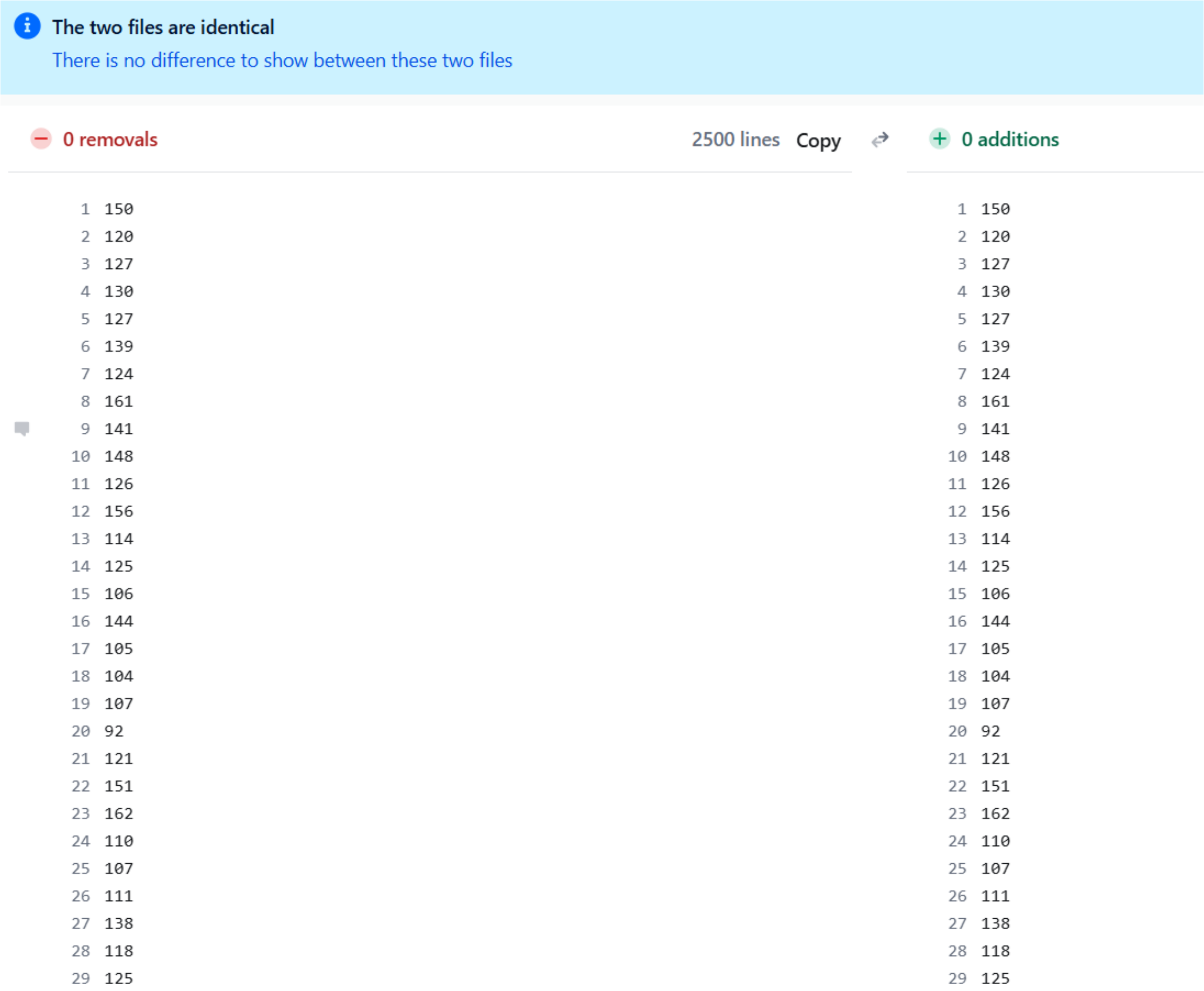
Maddie Masiello: Wrote a few test cases, helped verify correctness

Source Code:

```
#--------------------------------------------------------------------
# Lab 1 Code
#--------------------------------------------------------------------
.data
A: .word 1,2,3,4
B: .word 4,3,2,1
size: .word 2
C: .space 16
#--------------------------------------------------------------------
.text
Main:
# Initializing
li sp ,0x10000 # stack pointer
la s8, A # A Matrix
la s9, B # B Matrix
la s6, C # Mult C Matrix
la s5, size # Size of Matrix
li s1,0     # j counter
li s2,0     # i counter
li s3,0     # k counter
lw s4,0(s5) # Loads the value of size
mv s0,ra    # moves return address into saved reg
#--------------------------------------------------------------------
I_Loop: beq s2, s4, I_After # branch if i equals the value of size
      li s1,0      # reset j to 0

J_Loop: beq s1,s4, J_After # branch if j equals the value of size
      li s7,0     # clearing accumulator fo next iteration
      li s3,0     # reset k to 0

K_Loop: beq s3,s4,K_After # branch if k equals the value of size
        # getting the value in A matrix
      mv a1, s4   # moving save size reg to arg reg
mv a0, s2    #moving save i reg to arg reg

 # stack saving for a or t

  addi  sp,sp ,-8 # loading stack :(
  sw a0 ,0(sp)  # saving argumnet 0
  sw a1, 4(sp)  # saving argument 1
 call Multiply  # i * size
 mv t5,a0    # (i*size)
  lw a0 , 0(sp) # restoring argument 0
  lw a1, 4(sp) # restoring argumnet 1
  addi  sp,sp ,8 # restore the stack


 add t5,t5,s3 # (i*size)+k
 slli t5,t5,2 # ((i*size)+k) * 4
 add t2,s8,t5 # indexing A[((i*size)+k) * 4]
 lw s11,0(t2) # value at A[((i*size)+k) * 4]

 # getting the value in the B matrix
 mv a0,s3     # k into arg reg
 mv a1,s4     # size into arg reg

   #caller stack saving a0 and a1
  addi  sp,sp ,-8 # loading stack :(
  sw a0 , 0(sp)   #saving arg 0
  sw a1, 4(sp)    #saving arg 1
 call Multiply  # (k*size)
  mv t6,a0      # k *size
  lw a0 , 0(sp)  #restoring arg 0
  lw a1, 4(sp)   #restoring arg 1
  addi  sp,sp ,8 #restore the stack

  add t6,t6,s1  # (k*size)+j  --- k is t6
  slli t6,t6,2  # ((k*size)+j)*4
  add t4 ,s9,t6 # indexing B[(k*size)+j)*4]
  lw s10,0(t4)  # value at B[(k*size)+j)*4]


  #Multiplying Matrices values into the C matrices
  mv a1,s10     # value in B matrix
  mv a0,s11     # value in A matrix

  addi sp,sp,-8 # loading stack :(
  sw a0 , 0(sp)   # saving arg 0
  sw a1, 4(sp)     # saving arg 1
```

```
       call Multiply # Multiplying matrices values
       add s7,s7,a0 # accumualting products for each index in C
      lw a0 , 0(sp)   # restoring arg 0
      lw a1, 4(sp)     # restoring arg1
      addi sp,sp,8 # restore the stack

      # should now have multiplied  value

      addi s3,s3,1 # increment k by 1
      j K_Loop

K_After:
       addi s1,s1,1 # increment j by 1
       sw s7,0(s6) # store value into C matrix
       addi s6,s6,4  # incrementing the address of C for the new
value
       j J_Loop
J_After:
       addi s2,s2,1  # incrementing the i value
       j I_Loop

I_After:
       j Done
#---------------------------------------------------------------------
# The Multiply Function for Matrices
#---------------------------------------------------------------------
Multiply:
       li t6,0
M_Loop:
        beqz a0, Return # branch to return if a0 = 0
        andi t5,a0,1    # get the LSB
        beqz t5,Shifting # when LSB of B is 0 skip the adding
Adding:
        add t6,t6,a1  # adding to the multiply accumulator
Shifting:
        slli a1,a1,1  # shift A left once
        srli a0,a0,1  # shift B right once
        j M_Loop
Return:
       mv a0,t6 # put the return value in a0
       ret
#---------------------------------------------------------------------
Done:
       mv ra,s0
      #ret # comment out so it doesn't keep looping
```

Proof of working implementation:



The two files are identical

There is no difference to show between these two files

```
150        150
120        120
127        127
130        130
127        127
139        139
124        124
161        161
141        141
148        148
126        126
156        156
114        114
125        125
106        106
144        144
105        105
104        104
107        107
92         92
121        121
151        151
162        162
110        110
107        107
111        111
138        138
118        118
125        125
```