mbclausen@csu.fullerton.edu
youssefh@csu.fullerton.edu
jayson.doty@csu.fullerton.edu

Michael Clausen
Youssef Hegazy
Jayson Doty
CPSC 479
Dr. Bein
Dec. 6, 2023

Project 2 Report

Summary:

For this second project, we implemented a parallel Shear Sort algorithm using CUDA. Shear sort is a parallel sorting algorithm used to sort a 2-dimensional array of elements by alternately sorting the rows and columns of the matrix. CUDA is a platform and API made by NVIDIA to allow developers to use their GPUs for general purpose programming. Our objective was to use the parallel processing capabilities of an NVIDIA GPU and their CUDA interface to efficiently sort large, square matrices. This report includes the algorithm's pseudocode, our implementation in C/C++, and instructions on how to execute the program.

Pseudocode:

```
ShearSortRow function:
  Calculate the threadID based on the block and thread indices.
  If (threadID < matrix.size(n)):
    Loop over the matrix rows (i from 0 to n):
      Loop over the matrix columns (j from 0 to n-1):
        Calculate the index in the matrix based on the threadID and column index.
        If ( ( row%2==0 && element[index] > element[index+1)] ) or (row %2==1 &&
element[index] < element[index+1]) )
          swap the elements

ShearSortColumn function:
  Calculate the threadID based on the block and thread indices.
  If (threadID < matrix.size(n)):
    Loop over the matrix rows (i from 0 to n):
      Loop over the matrix columns (j from 0 to n-1):
        Calculate the index in the matrix based on the column index and threadID.
        If ( element[index] > element[index+1] )
          swap the elements.
```

Main function:

```
Initialize N to 0.
Display a prompt for the user to enter an integer value for N greater than 0.
Get the value of N from user input.
While (N < 1):
        Display a message that N must be greater than 0.
        Get the value of N from user input again.

Declare a 2D array 'ha' with dimensions N x N.
Declare pointers 'dm' and 'd_count' for GPU memory.

// Initialize the matrix with random values and print it.
Display "Matrix before sorting" with a separator line.
For each row (i from 0 to N):
        For each column (j from 0 to N):
        Assign a random value to ha[i][j].
        Display the value of ha[i][j] with proper formatting.

// Memory allocation and data transfer to GPU.
Allocate memory on GPU for 'dm' with size N x N.
Copy the matrix 'ha' from host to device memory.

Define the number of threads per block and the number of blocks.
Set threadsPerBlock to N and numBlocks to 1.

For (i from 0 to log2(N) + 1):
        Launch the ShearSortRow kernel function with parameters 'dm' and 'N'.
        Synchronize the device to ensure kernel execution is completed.
        Launch the ShearSortColumn kernel function with parameters 'dm' and 'N'.
        Synchronize the device again.

// Copy the sorted matrix back to the host and free GPU memory.
Copy the matrix 'dm' from device to host memory.
Free the memory allocated for 'dm' on the GPU.

// Print the sorted matrix.
Display "Matrix After sorting" with a separator line.
For each row (i from 0 to N):
        For each column (j from 0 to N):
        Display the value of ha[i][j] with proper formatting.
```

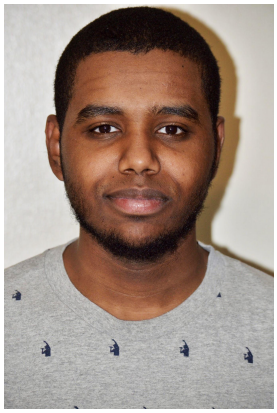
How to Compile and Run the Program:

1. On a system that has the NVIDIA Cuda Compiler Driver installed, compile the program in the command prompt: **nvcc Project2.cu**
2. Run the program by providing the compiled file name to the command prompt, by default: **./a.out**
3. The program will ask the user to input a positive integer N that will be used to create an NxN matrix filled with random values.
4. The program will sort the matrix using a shear sort algorithm. The generated matrix will be displayed before and after the sorting is performed.

Screenshots:

Jayson Doty

Michael Clausen



Youssef Hegazy

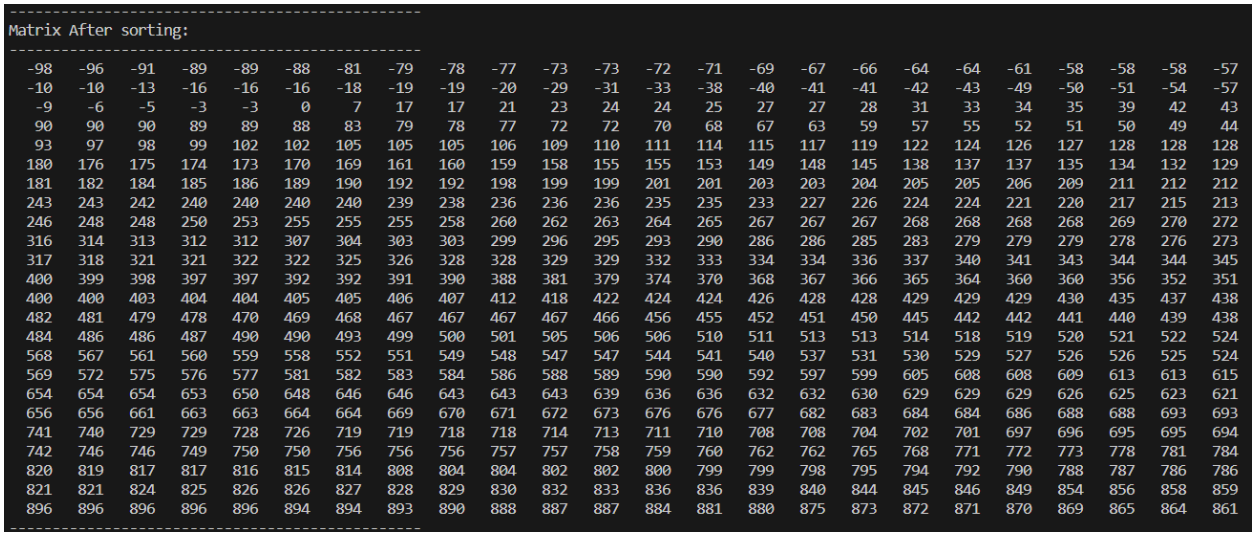
```
jayson.doty@prudence:~/project-2-team_jayson/project-2-team_jayson$ nvcc Project2.cu
Project2.cu(72): warning: variable "d_count" was declared but never referenced

jayson.doty@prudence:~/project-2-team_jayson/project-2-team_jayson$ ./a.out
Enter an integer value for n greater than 0: 16
Matrix before sorting:
-------------------------------------------------
  283   786   677   815   693   235   286   392   549   321   262   -73   590   -41   663   826
  440   326    72   636   111   268   467   329   682   430   762    23   -33    35   829   702
  -78   -42   -31    67   293   356   -89   -58   129   273   321   819   684   437    98   224
  215   270   313   426    -9   880   856   773   762    70   896   181   205   825   -16   227
  236   405   746   629   213   757    24   795   482   445   714   267   334   264   -57   650
  -13   708   176    78   688   484   303   551   654   299   832   -40   576   268   639   -88
  126   486    -6   439   695   470   334   278   367   501    -3   802   217   392   552   656
  201   180   186   341   765   589   344   519   340   629   -69    17    -3   671   381   575
  609   827   467   756   397   253   486   865   206   583   119   524   428   771   632   729
  403   -81   170   268   608   615   240    49   696   623   518   145   746   351   821   455
  279   388   664   128   741   250    93   400   -66   664    24   814   887   756   643   391
  127   265   759   836   332   451   337   128   175   307   374    21   758   295   -71   137
  135   693   718   328    43   -89   828   429   676   304   343   663   513   438   506   740
  804   718    28   588   269   817   817   896   224   643   370    83   390   399   672   625
  544   490   405    39   854   686   569   -18   442   364    97   407   255   704   248   511
  522   728   199   243   646   468   240   322   211   710   505   701   561   630   778   205
-------------------------------------------------
Matrix After sorting:
-------------------------------------------------
  -89   -89   -88   -81   -78   -73   -71   -69   -66   -58   -57   -42   -41   -40   -33   -31
   43    39    35    28    24    24    23    21    17    -3    -3    -6    -9   -13   -16   -18
   49    67    70    72    78    83    93    97    98   111   119   126   127   128   128   129
  213   211   206   205   205   201   199   186   181   180   176   175   170   145   137   135
  215   217   224   224   227   235   236   240   240   243   248   250   253   255   262   264
  303   299   295   293   286   283   279   278   273   270   269   268   268   268   267   265
  304   307   313   321   321   322   326   328   329   332   334   334   337   340   341   343
  400   399   397   392   392   391   390   388   381   374   370   367   364   356   351   344
  403   405   405   407   426   428   429   430   437   438   439   440   442   445   451   455
  519   518   513   511   506   505   501   490   486   486   484   482   470   468   467   467
  522   524   544   549   551   552   561   569   575   576   583   588   589   590   608   609
  663   656   654   650   646   643   643   639   636   632   630   629   629   625   623   615
  663   664   664   671   672   676   677   682   684   686   688   693   693   695   696   701
  757   756   756   746   746   741   740   729   728   718   718   714   710   708   704   702
  758   759   762   762   765   771   773   778   786   795   802   804   814   815   817   817
  896   896   887   880   865   856   854   836   832   829   828   827   826   825   821   819
-------------------------------------------------
```

Image 1: Screenshot of a 16x16 matrix being sorted

```
Enter an integer value for n greater than 0: 24
Matrix before sorting:
------------------------------------------------
  283   786   677   815   693   235   286   392   549   321   262   -73   590   -41   663   826   440   326    72   636   111   268   467   329
  682   430   762    23   -33    35   829   702   -78   -42   -31    67   293   356   -89   -58   129   273   321   819   684   437    98   224
  215   270   313   426    -9   880   856   773   762    70   896   181   205   825   -16   227   236   405   746   629   213   757    24   795
  482   445   714   267   334   264   -57   650   -13   708   176    78   688   484   303   551   654   299   832   -40   576   268   639   -88
  126   486    -6   439   695   470   334   278   367   501    -3   802   217   392   552   656   201   180   186   341   765   589   344   519
  340   629   -69    17    -3   671   381   575   609   827   467   756   397   253   486   865   206   583   119   524   428   771   632   729
  403   -81   170   268   608   615   240    49   696   623   518   145   746   351   821   455   279   388   664   128   741   250    93   400
  -66   664    24   814   887   756   643   391   127   265   759   836   332   451   337   128   175   307   374    21   758   295   -71   137
  135   693   718   328    43   -89   828   429   676   304   343   663   513   438   506   740   804   718    28   588   269   817   817   896
  224   643   370    83   390   399   672   625   544   490   405    39   854   686   569   -18   442   364    97   407   255   704   248   511
  522   728   199   243   646   468   240   322   211   710   505   701   561   630   778   205   220   636   344   526   422   365   608   316
  182   158   824   537   -38   524   500   -64   352   799   279   450   368   -29   873    31   781   830   833   794   560    63    99   881
  799   896   859   673   713   568    90    -5   826   366   -16   240   -10   584   276   442   836     7   345   656    79   318   787   312
  248    72   559   -91   236   110   242   487   106   201   613   272   221   155   719   499   621   804   839   711   840   567   605   128
   27    50   884   558   820   124   322   169   296   -19   530   -16   192   872   572   750   525   285   122   199   540   -58   798   613
  198    90   424   490   109   481   719   236   632    55   894   -96   279   669   173   676   750   155   760    42   479   784   893   105
  521   467   404   513   861   654   226   159   844   102   102   406   684   -79   742   768   428    89   772   808   858   398   -64   708
  653   148   203   233    33   548   790   654   467   646   268   429   400   -54   688   697   149   890   203   -67   263   397   153   792
  586    25    52   896   875    88    57   629   336   360   314   821   360   204   -72   -73   -50   648   456   802   694   597   599   -57
  -61   -98   328   303   400   581   547   438    59    51   435    34   239   592   115    27   404   529   -51   864   185   329   243   235
   77   800   138   871   849   189   267   888   192   695   643    44   729   290   582   240   441   469   726   132   161   -58   260    17
  -77   661   -19   209    90   325   896   267   577   134   590   526   424   -43   514    68   105   258   212   286     0   246   626   894
  816   452   478   429   846   190   547   870   -49   -20   531   493   757   527   212   786   114   255   412   -10   312   379   510   869
   89   174   255   541   520   333   887   788   238   466   670   184   756   317   506   160   749   137   105   -41   117   418   845   683
```

Image 2: A 24x24 matrix before shear sort.



```
------------------------------------------------
Matrix After sorting:
------------------------------------------------
 -98  -96  -91  -89  -89  -88  -81  -79  -78  -77  -73  -73  -72  -71  -69  -67  -66  -64  -64  -61  -58  -58  -58  -57
 -10  -10  -13  -16  -16  -16  -18  -19  -19  -20  -29  -31  -33  -38  -40  -41  -41  -42  -43  -49  -50  -51  -54  -57
  -9   -6   -5   -3   -3    0    7   17   17   21   23   24   24   25   27   27   28   31   33   34   35   39   42   43
  90   90   90   89   89   88   83   79   78   77   72   72   70   68   67   63   59   57   55   52   51   50   49   44
  93   97   98   99  102  102  105  105  105  106  109  110  111  114  115  117  119  122  124  126  127  128  128  128
 180  176  175  174  173  170  169  161  160  159  158  155  155  153  149  148  145  138  137  137  135  134  132  129
 181  182  184  185  186  189  190  192  192  198  199  199  201  201  203  203  204  205  205  206  209  211  212  212
 243  243  242  240  240  240  240  239  238  236  236  236  235  235  233  227  226  224  224  221  220  217  215  213
 246  248  248  250  253  255  255  255  258  260  262  263  264  265  267  267  267  268  268  268  268  269  270  272
 316  314  313  312  312  307  304  303  303  299  296  295  293  290  286  286  285  283  279  279  279  278  276  273
 317  318  321  321  322  322  325  326  328  328  329  329  332  333  334  334  336  337  340  341  343  344  344  345
 400  399  398  397  397  392  392  391  390  388  381  379  374  370  368  367  366  365  364  360  360  356  352  351
 400  400  403  404  404  405  405  406  407  412  418  422  424  424  426  428  428  429  429  429  430  435  437  438
 482  481  479  478  470  469  468  467  467  467  467  466  456  455  452  451  450  445  442  442  441  440  439  438
 484  486  486  487  490  490  493  499  500  501  505  506  506  510  511  513  513  514  518  519  520  521  522  524
 568  567  561  560  559  558  552  551  549  548  547  547  544  541  540  537  531  530  529  527  526  526  525  524
 569  572  575  576  577  581  582  583  584  586  588  589  590  590  592  597  599  605  608  608  609  613  613  615
 654  654  654  653  650  648  646  646  643  643  643  639  636  636  632  632  630  629  629  629  626  625  623  621
 656  656  661  663  663  664  664  669  670  671  672  673  676  676  677  682  683  684  684  686  688  688  693  693
 741  740  729  729  728  726  719  719  718  718  714  713  711  710  708  708  704  702  701  697  696  695  695  694
 742  746  746  749  750  750  756  756  756  757  757  758  759  760  762  762  765  768  771  772  773  778  781  784
 820  819  817  817  816  815  814  808  804  804  802  802  800  799  799  798  795  794  792  790  788  787  786  786
 821  821  824  825  826  826  827  828  829  830  832  833  836  836  839  840  844  845  846  849  854  856  858  859
 896  896  896  896  896  894  894  893  890  888  887  887  884  881  880  875  873  872  871  870  869  865  864  861
------------------------------------------------
```

Image 3: The same 24x24 matrix after shear sort.