Assignment 3

Stuart Hopkins, A02080107

January 2020

1 Integer Sort

To create a sorting program I start with process 0. It randomly generates an array of numbers to the specified size. Process 0 then sends random sized segments of the array to the slave processes. These slave processes sort their part using the built in sort function.

As process 0 receives all the segments back it merges them together. After merging them together it prints out the final array. The program then terminates. My code is as follows:

```
#include <iostream>
#include <mpi.h>
#include <unistd.h>
#include <stdlib.h>
#include <time.h>
#include <algorithm>

using namespace std;

#define MCW MPI_COMM_WORLD

void printArray(int arr[], int size, string start_message, string end_message);

int main(int argc, char **argv){

  int rank, size;
  const int ARR_SIZE = 64;
  int data[ARR_SIZE];
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MCW,&rank);
  MPI_Comm_size(MCW,&size);
  MPI_Status mystatus;

  // Use process 0 as the master process
  if(!rank) {
    srand(time(NULL));

    // Initialize the array
    // Start at 1 because this process is already rank 0
```

```
  for(int i = 0; i < ARR_SIZE; i++) {
    data[i] = rand() % ARR_SIZE;
  }
  printArray(data, ARR_SIZE, "<<<Beginning Array>>>", "");

  // Send out random chunks to processes
  int size_left = ARR_SIZE;
  int TMP_ARR_SIZE;
  for(int i = 1; i < size; i++) {
    if(!size_left) break;


    if(i+1 != size) {
      int prev_size_left = size_left;
      size_left -= rand() % size_left;
      TMP_ARR_SIZE= prev_size_left - size_left;
      MPI_Send(&data[size_left], TMP_ARR_SIZE, MPI_INT, i, 0, MCW);
    }
    else {
      MPI_Send(&data[0], size_left, MPI_INT, i, 0, MCW);
    }
  }
}

// Work processes sort array
else {
  MPI_Probe(MPI_ANY_SOURCE, 0, MCW, &mystatus);
  int count;
  MPI_Get_count(&mystatus, MPI_INT, &count);
  MPI_Recv(data, count, MPI_INT, MPI_ANY_SOURCE, 0, MCW, &mystatus);

  sort(&data[0], &data[count]);
  // printArray(data, count, "BEGIN " + to_string(rank), "END");
  MPI_Send(data, count, MPI_INT, 0, 0, MCW);
}

// Combine sorted arrays
if(!rank) {
  int new_arr[ARR_SIZE];
  int pos = 0;

  for(int i = 1; i < size; i++) {
    MPI_Probe(i, 0, MCW, &mystatus);
    int count;
    MPI_Get_count(&mystatus, MPI_INT, &count);
    MPI_Recv(data, count, MPI_INT, MPI_ANY_SOURCE, 0, MCW, &mystatus);

    int prev_pos = pos;
```

```
for(int i = 0; i < count; i++) {
  new_arr[pos] = data[i];
  pos++;
}

// Merge
int size_a = prev_pos - 0;
int size_b = pos - prev_pos;
int size_total = pos - 0;

int tmp_arr[size_total];

int a_pos = 0;
int b_pos = prev_pos;
int tmp_pos = 0;

while(true) {
  if(a_pos >= size_a) {
    while(b_pos <= pos) {
      tmp_arr[tmp_pos] = new_arr[b_pos];
      tmp_pos++;
      b_pos++;
    }
    break;
  }

  else if(b_pos >= pos) {
    while(a_pos < prev_pos) {
      tmp_arr[tmp_pos] = new_arr[a_pos];
      tmp_pos++;
      a_pos++;
    }
    break;
  }

  else if(new_arr[a_pos] < new_arr[b_pos]) {
    tmp_arr[tmp_pos] = new_arr[a_pos];
    tmp_pos++;
    a_pos++;
  }

  else {
    tmp_arr[tmp_pos] = new_arr[b_pos];
    tmp_pos++;
    b_pos++;
  }
}
for(int i = 0; i < ARR_SIZE; i++) {
```

```
        new_arr[i] = tmp_arr[i];
      }
    }

    printArray(new_arr, ARR_SIZE, "<<<Sorted Array>>>", "");
  }

  MPI_Finalize();
}


void printArray(int arr[], int size, string start_message, string end_message) {
  cout << start_message << endl;
  for(int i = 0; i < size; i++) {
    cout << arr[i] << " ";
  }
  cout << endl << end_message << endl;
}
```

To verify it works here is the command line arguments and output It worked, here is the code:

```
$ mpic++ main.cpp
$ mpirun -np 4 a.out

<<<Beginning Array>>>
46 10 63 41 2 11 45 40 50 58 39 1 52 53 60 40 43 0 51 5 3 3
23 27 37 48 61 5 43 20 62 26 30 61 3 32 9 48 8 59 42 47 60
31 37 56 7 16 57 59 22 60 62 45 23 35 29 20 40 9 40 39 35 6

<<<Sorted Array>>>
0 1 2 3 3 3 5 5 6 7 8 9 9 10 11 16 20 20 22 23 23 26 27 29
30 31 32 35 35 37 37 39 39 40 40 40 40 41 42 43 43 45 45 46
47 48 48 50 51 52 53 56 57 58 59 59 60 60 60 61 61 62 62 63
```

Notice that you can set the length of the array at the top of the program. This output is for the current max of 64.