Evan Nichols
EECS 678
Lab 8 Report

My complete code with comments.

```
/*
 * Job struct is used to store data on each job
 */
typedef struct {
  int id;
  int arrival_time;
  int prio;
} job;

/*
 * Array to hold a max of ten job structs
 */
 job job_arr[10];

/*
 * File reading to process the process.txt input file.
 */
 void readFile(char* myfile){
   FILE *fr;
   int counter = 0;
   int id, arrive, prio;

   fr = fopen(myfile, "r");

   //consume each line saving each var to the appropriate struct field
   while( fscanf(fr,"%d %d %d",&id,&arrive,&prio) == 3 ){
        job_arr[counter].id = id;
        job_arr[counter].arrival_time = arrive;
        job_arr[counter].prio = prio;
        counter++;
   }
   fclose(fr);
}

/*
 * Arrival Comparer function. Jobs sorted in ascending order
 * based on their arrival time.
 */
int arrival_comparer(const void *j1, const void *j2){
       //cast the void points to job pointer to access their data
       job job1 = *((job *)j1);
       job job2 = *((job *)j2);

       return job1.arrival_time - job2.arrival_time;
}

/*
```

```
 * Priority Comparer function. Jobs sorted in descending order. If two
 * priorities are equal, arrival time is considered.
 */
int prio_comparer(const void * j1, const void * j2){
	job job1 = *(job *)j1;
	job job2 = *(job *)j2;

	if(job1.prio == job2.prio){
	return job1.arrival_time - job2.arrival_time;
	}
	return job1.prio - job2.prio;
}

/*
 * Helper method to print out all jobs stored in the array
 */
void print_jobs(job arr[], int size){
	for(int i = 0; i < size; i++){
	printf("JOB ID: %d, ARRIVAL: %d, PRIO: %d
\n",arr[i].id,arr[i].arrival_time,arr[i].prio);
	}
	printf("\n");
}

int main(int argc, char **argv)
{
	/*
	* Read input file and print out the initial jobs list
	*/
	readFile("process.txt");
	printf("BEFORE SORTING\n\n");
	print_jobs(job_arr, 7);

	void *ptr = job_arr;

	/*
	* Declare compare functions
	*/
	int (*arrivalcomp)(const void*, const void*) = arrival_comparer;
	int (*priocomp)(const void*, const void*) = prio_comparer;

	qsort(ptr, 7, sizeof(job),arrivalcomp);
	printf("AFTER SORTING BY ARRIVAL TIME\n\n");
	print_jobs(job_arr, 7);

	qsort(ptr, 7, sizeof(job),priocomp);
	printf("AFTER SORTING BY PRIORITY\n\n");
	print_jobs(job_arr, 7);

	return 0;
}
```