**Sorting Algorithms**

**1 Comparison Sorts**

You are consulting for a small restaurant chain which wants to expand its locations nationwide. The company executives would like to start their expansion by placing locations in the most densely populated areas ﬁrst. To assist them, you have been asked to produce a list of Census Tracts ordered by population density. You have been given a list of the approximately 74,000 Census Tracts in the United States. The name, area (in square kilometers), total population, and population density (in people per square kilometer) are given to you for each tract. Your job will be to write a program which reads this data, sorts the tracts by population density, and outputs all of this data sorted by population density (from smallest to largest).

**Input File Format:**

File Name: **pop\_density.txt**

Each record of the input ﬁle is on a separate line. Each line contains (in this order) a name (string), an area in square kilometers (double), a population count (int), and a population density (double).

A pipe ( | ) is used to separate the diﬀerent attributes of each record.

Below are three example records from the input ﬁle:

Census Tract 201, Autauga County, Alabama|9.84473419420788|1808|183.651479494869

Census Tract 202, Autauga County, Alabama|3.34583234555866|2355|703.860730836106

Census Tract 203, Autauga County, Alabama|5.35750339330735|3057|570.60159846447

**Sorting Algorithm Requirements:** For this project, you will implement two diﬀerent sorting techniques and examine their performances: MergeSort and QuickSort. Do not use a C++ library for the sorting. You must write own code to perform the sorting operations. For QuickSort, choose the pivot using the median-of-three approach. You must write a separate program for each of these two sorting techniques. Your programs should track and output the execution time for each sorting method. Clock provides a simple mechanism for timing code using clicks. Only time the sort routines, not the entire program.

**Program Input/Output Requirements:** Each of your two programs must create a new ﬁle named **pop\_density\_sorted.txt** which contains the records from the input ﬁle sorted in ascending order of population density. The format of the output ﬁle must be identical to that of the input ﬁle. The only diﬀerence between the input ﬁle and output ﬁle will be the order of the records. In other words, each record must still contain the name, area, population count, and population density. Each of your two programs must also print the number of clicks taken by the sorting procedure. All programs must be written in C++. Please comment your code appropriately.

**2 Integer Sorts**

Write a program that reads data from the given input ﬁle **integers.txt**, sorts them in ascending order using Radix Sort and writes the sorted list in another ﬁle named **integers sorted.txt**