```
//Created by Nathan Gaffney with functions from class
#include <iostream>
#include <fstream>
#include "Employee.h"
void bubleSortEmployee(EmployeeData anArray[], int numElems);
void sinkSortEmployee(EmployeeData anArray[], int numElems);
void populateArray(std::ifstream &inFile, EmployeeData anArray[], int& numElems);
void outputArray(std::ofstream &outFile, EmployeeData anArray[], int numElems);
void addElement(EmployeeData anArray[], int numElems);
void deleteElement(EmployeeData anArray[], int& numElems);
int main()
{
	const int MAX_ARRAY = 25;
	int numElems, choice = 0;
	EmployeeData  empArray[MAX_ARRAY];
	bool cont = true;
	std::ifstream inFile("EmployeeInfo.txt");
	std::ofstream outFile("SortedArray.txt");
	populateArray(inFile, empArray, numElems);
	do
	{
		std::cout << "What would you like to do" << std::endl <<
			"1. Add an element." << std::endl <<
			"2. Delet an Element" << std::endl <<
			"3. Bubble Sort" << std::endl <<
			"4. Sink Sort" << std::endl <<
			"5. Output the Array" << std::endl <<
			"0. To Exit." << std::endl;
		std::cin >> choice;
		switch (choice)
		{
		case 1:
			addElement(empArray, numElems);
			break;
		case 2:
			deleteElement(empArray, numElems);
			break;
		case 3:
			bubleSortEmployee(empArray, numElems);
			break;
		case 4:
			sinkSortEmployee(empArray, numElems);
			break;
		case 5:
			outputArray(outFile, empArray, numElems);
			break;
		default:
			cont = false;
			break;
		}
	} while (cont);
	std::system("pause");
	return 0;
}
void bubleSortEmployee(EmployeeData anArray[], int numElems)
{
	int            placePtr;    // Index for comparing adjacent elements
	int            endIndex;    // Index of stopping point for one pass
```

```
	EmployeeData	temp;		// Needed for swapping contents

	for (endIndex = numElems - 1; endIndex >= 0; endIndex--)
	{
		for (placePtr = 0; placePtr <= endIndex; placePtr++)
			if (anArray[placePtr].getHoursWorked() < anArray[placePtr +
1].getHoursWorked())
			{
			temp.setHoursWorked(anArray[placePtr + 1].getHoursWorked());
			temp.setPayPerHour(anArray[placePtr + 1].getPayPerHour());
			temp.setIdNumber(anArray[placePtr + 1].getIdNumber());
			anArray[placePtr +
1].setHoursWorked(anArray[placePtr].getHoursWorked());
			anArray[placePtr +
1].setPayPerHour(anArray[placePtr].getPayPerHour());
			anArray[placePtr + 1].setIdNumber(anArray[placePtr].getIdNumber());
			anArray[placePtr].setHoursWorked(temp.getHoursWorked());
			anArray[placePtr].setPayPerHour(temp.getPayPerHour());
			anArray[placePtr].setIdNumber(temp.getIdNumber());
			}
	}
}
void sinkSortEmployee(EmployeeData anArray[], int numElems)
{
	int		placePtr;	// Index for comparing adjacent elements
	int		endIndex;	// Index of stopping point for one pass
	EmployeeData	temp;		// Needed for swapping contents

	for (endIndex = 0; endIndex <= numElems; endIndex++)
	{
		for (placePtr = 0; placePtr <= endIndex; placePtr++)
			if (anArray[placePtr].getPayPerHour() > anArray[placePtr +
1].getPayPerHour())
			{
			temp.setHoursWorked(anArray[placePtr + 1].getHoursWorked());
			temp.setPayPerHour(anArray[placePtr + 1].getPayPerHour());
			temp.setIdNumber(anArray[placePtr + 1].getIdNumber());
			anArray[placePtr +
1].setHoursWorked(anArray[placePtr].getHoursWorked());
			anArray[placePtr +
1].setPayPerHour(anArray[placePtr].getPayPerHour());
			anArray[placePtr + 1].setIdNumber(anArray[placePtr].getIdNumber());
			anArray[placePtr].setHoursWorked(temp.getHoursWorked());
			anArray[placePtr].setPayPerHour(temp.getPayPerHour());
			anArray[placePtr].setIdNumber(temp.getIdNumber());
			}
	}
}
void populateArray(std::ifstream &inFile, EmployeeData anArray[], int& numElems)
{
	std::string id;
	double hours;
	double pay;
	int ptr = 0;
	inFile >> id >> hours >> pay;		// Priming read
	while (!inFile.eof())
	{
		// Store information from file in current array object
```

```
			anArray[ptr].setIdNumber(id);
			anArray[ptr].setHoursWorked(hours);
			anArray[ptr].setPayPerHour(pay);

			inFile >> id >> hours >> pay;	// Continuation read
			ptr++;
		}
		numElems = ptr;
}
void outputArray(std::ofstream &outFile, EmployeeData anArray[], int numElems)
{
		for (int i = 0; i < numElems; i++)
		{
			outFile << anArray[i].toString();

		}
}
void addElement(EmployeeData anArray[], int numElems)
{
		double hours, worked;
		std::string id;
		int place = numElems + 1;
		std::cout << "Enter the employee Id Number: ";
		std::cin >> id;
		std::cout << "Enter the number of hours worked: ";
		std::cin >> hours;
		std::cout << "Enter the amount paid per hour: " << std::endl;
		std::cin >> worked;
		anArray[place].setIdNumber(id);
		anArray[place].setHoursWorked(hours);
		anArray[place].setPayPerHour(worked);
}
void deleteElement(EmployeeData anArray[], int& numElems)
{
		double oldNum;
		int ptr;
		std::cout << "Enter the hours worked by the employee: ";
		std::cin >> oldNum;
		ptr = 0;				// Scan list for deletion target
		while (oldNum != anArray[ptr].getPayPerHour() && ptr < numElems)
			ptr++;

		if (ptr < numElems)			// If target found, then
		{
			anArray[ptr].setHoursWorked(anArray[numElems - 1].getHoursWorked());
			anArray[ptr].setPayPerHour(anArray[numElems - 1].getPayPerHour());
			anArray[ptr].setIdNumber(anArray[numElems - 1].getIdNumber());
			numElems--;			// Decrement size of list
		}
		bubleSortEmployee(anArray, numElems);
}
```

```
AB123456 32  29.85
DF634581 14  4.66
ZG196482 40  20.99
HR463237 45  12.56
YU765157 29  32.14
IJ785243 40  24.16
```

```cpp
//Created by: Nathan Gaffney
//Attempting to make my own format
//This class holds information on an employee
#include <iostream>
#include <string>
class EmployeeData
{
private:
	double hoursWorked;
	double payPerHour;
	std::string idNumber;

public:
	/*************Basic Class Functions***************/
	//Constructor
	EmployeeData();
	//Getters and Setters for class attributs
	void	setHoursWorked(double a) { hoursWorked = a; }
	double	getHoursWorked()	 { return hoursWorked; }

	void		setPayPerHour(double a) { payPerHour = a; }
	double	getPayPerHour()	 { return payPerHour; }

	void	setIdNumber(std::string a) { idNumber = a; }
	std::string getIdNumber()	 { return idNumber; }

	std::string toString();

};
EmployeeData::EmployeeData()
{
	hoursWorked = -1;
	payPerHour = -1;
	idNumber = "IdNumber was not set";
}
std::string EmployeeData::toString()
{
	std::string output;
	std::string space = "   ";
	output = idNumber + space + std::to_string(hoursWorked) + space +
std::to_string(payPerHour) + "\n";
	return output;
}
```

Output:

```
HR463237   45.000000   12.560000
ZG196482   40.000000   20.990000
IJ785243   40.000000   24.160000
AB123456   32.000000   29.850000
YU765157   29.000000   32.140000
DF634581   14.000000   4.660000
```