Source Code:

EmployeeClass.cpp

```
/**
*TODO:The purpose of this program is to keep track of employee information such as name, ID
number, Department, and position
@param name - Sheharyar Khan
@param courseInfo - CS-116
@return - none
*/
#include <vector>//including vector library to use vectors
#include <iostream>
#include "EmployeeHeader.hpp" //using a header file for the class and the functions
#include "printmefirst.hpp" //using a separate header file for the print me first function as
it will probably be used in every project for this course and this would be easier than
copying the code over and over again. I also wanted to apply what I learned today in class
#include <sstream>//using the stringstream library for stringstream objects
#include <fstream> //using fstream library to read and write to files

using namespace std;

int main()
{
printMeFirst ("Sheharyar Khan", "CS - 116: Lab 2"); //put your name instead of “Ron Sha”
// Create an Employee object
Employee X;
//define character that splits data in the text file
const char SPLIT_CHAR = ';';
//define file path
string fileName="Employee.txt";
//declare vector
vector <Employee> employeeList;
//declaring Substring
string subString;
//declaring input file stream
ifstream textFile;
textFile.open(fileName);
//if the file is open then run the loop else tell the user the file failed to open
if(textFile.is_open())
{
   while(textFile.good())
   {//in the text file get a substring till the end of line character
       getline(textFile, subString);
       string stringArray [4];
       int i=0;
       stringstream ss(subString);
       while(ss.good())
       {   string substr;
       //in the substring get a sub-substring till the splitting character
           getline(ss,substr,SPLIT_CHAR);
           //add sub-substring to array
           stringArray[i]=substr;
           i++;
       }
       //convert string to int by using >> operator, basically using cin to define the
variable but instead of the console input we're using a string
       stringstream idNum(stringArray[1]);
       int idNumInt;
       idNum >> idNumInt;
       //calling operater constructor and creating employee x
       Employee X(stringArray[0],idNumInt,stringArray[2],stringArray[3]);
       //adding employee x to the vector
       employeeList.push_back(X);
   }
   //couting list headings and dividers
   cout<<"Name \t \t ID \t Dept \t \t Position"<<endl;
```

```
   cout<<"---- \t \t -- \t ---- \t \t --------"<<endl;

   //couting all the employees in the vector
   for(int i =0;i<employeeList.size();i++)
   {
       cout<<employeeList[i]<<endl;
   }
}
else
{//if file fails to open then...
   cout<<"ERROR! The text file failed to open."<<endl;
   return -1;
}
return 0;
}
```

EmployeeHeader.cpp

```
#ifndef EMPLOYEE_H
#define EMPLOYEE_H

#include <iostream>
#include <string>

using namespace std;

class Employee
{
   private:    //member variables
   string name;
   int idNumber;
   string department;
   string position;

   public: //mutator and accessor functions
   //Parameters:    Name       idNumber          Department          Position
   //these parameters are assigned to the member variables
   Employee(string eName, int eIdNumber, string eDepartment, string ePostion);
   //Parameters:    Name       idNumber
   //these parameters are assigned to the member variables
   Employee(string eName, int eIdNumber);
   Employee();

   string getName();
   int getIdNumber();
   string getDepartment();
   string getPosition();

   void setName(string eName);
   void setIdNumber(int eIdNumber);
   void setDepartment(string eDepartment);
   void setPosition(string ePosition);
};
         //Parameters:    Name       idNumber          Department          Position
        //these parameters are assigned to the member variables
Employee::Employee(string eName, int eIdNumber, string eDepartment, string ePosition)
{
   name = eName;
   idNumber = eIdNumber;
   department = eDepartment;
   position = ePosition;
}
         //Parameters:    Name       idNumber
        //these parameters are assigned to the member variables
Employee::Employee(string eName, int eIdNumber)
{
   name = eName;
```

```
   idNumber = eIdNumber;
   department = "";
   position = "";
}

Employee::Employee()
{
   name = "";
   idNumber = 0;
   department = "";
   position = "";
}

//returns the name
string Employee::getName()
{
   return name;
}
//returns the idnumber
int Employee::getIdNumber()
{
   return idNumber;
}
//returns the department
string Employee::getDepartment()
{
   return department;
}
//returns the position
string Employee::getPosition()
{
   return position;
}
//sets the name. @param eName - gets the name and assigns it to the member variable name
void Employee::setName(string eName)
{
   name = eName; //simply sets the attribute to the input in the parameter
}
//sets the IdNumber. @param eIdNumber - gets the IdNumber and assigns it to the member
variable IdNumber
void Employee::setIdNumber(int eIdNumber)
{
   idNumber = eIdNumber;
}
//sets the Department. @param eDepartment - gets the Department and assigns it to the member
variable Department
void Employee::setDepartment(string eDepartment)
{
   department = eDepartment;
}
//sets the Position. @param ePosition - gets the Position and assigns it to the member
variable Position
void Employee::setPosition(string ePosition)
{
   position = ePosition;
}
//displays the Employee's Info by a series of cout statements
void displayEmployee(Employee employee)
/*this function is not part of the class because of the way it is called in the driver
function. The way its called suggests that it is function outside the class scope.*/
{
   cout<<"Name: "<<employee.getName()<<endl;
   cout<<"ID Number: "<<employee.getIdNumber()<<endl;
   cout<<"Department: "<<employee.getDepartment()<<endl;
   cout<<"Position: "<<employee.getPosition()<<endl<<endl;
}
```

```
//overloaded << operator
//I overloaded this operator for practice. it was not really needed as the displayEmployee
function could be overloaded or changed to produce the same result just by changing "endl;" to
a tab "\t".
//this function overloaded in the same way as the overloaded operator in my money class but a
brief summary would be:
//first I load everything into the output stream variable 'output' then return it to the cout
statement in the main code and that prints it out to the console.
//cout function
   std::ostream& operator<< (std::ostream& output, Employee &right)
       {
           //using an if else branch to align the list if one of the departments is too short
such as "IT" using "\t"
           if(right.getDepartment().length()<5){

output<<right.getName()<<"\t"<<right.getIdNumber()<<"\t"<<right.getDepartment()<<"\t
\t"<<right.getPosition();
           }else{

output<<right.getName()<<"\t"<<right.getIdNumber()<<"\t"<<right.getDepartment()<<"\t"<<right.g
etPosition();
           }
           return output;
       }

#endif
```

Printmefirst.hpp

```
#ifndef PRINTMEFIRST_H
#define PRINTMEFIRST_H

#include <iostream>
#include <ctime>
#include <iomanip>

using namespace std;

void printMeFirst(string name, string courseInfo)
{
cout <<" Program written by: "<< name << endl; // put your name here
cout <<" Course info: "<< courseInfo << endl;
time_t now = time(0); // current date/time based on current system
char* dt = ctime(&now); // convert now to string for
cout << " Date: " << dt << endl;
}

#endif
```

Employee.txt

```
Susan Meyers; 47889; Accounting; Vice President
Mark Jones; 39119; IT; Software Engineer
Joy Rogers; 81774; Manufacturing; Engineer
```

Purpose

The purpose is to make a program that correctly parses a text file, creates an object from the variables it parsed, add a that object to a vector and when it's done reading the file, creating object and adding it to the vector, it should print the contents of each of the objects on to the console.

Logic

The class is the same as the employee class from earlier other than the new overloaded '<<' operator which is the same as the overloaded '<<' operator in the Money Class lab from earlier.

The only new code in this lab is the parsing code. Basically what I'm doing is first reading a line from text file from start to the end of line character at the end of the line. I did this using the getline function. Next I broke that line up to 3 more parts using the overloaded getline function and entering the end of line character 'SPLIT_CHAR'. As I split the characters I assigned them to an Array in a loop. I'm using an array because it's easier to assign the variables in a loop. After all of the data has been transferred to the array I converted the IdNumber variable stored in the array at index [1] from a string to an int using basically cin but instead of using the console to input data to the int variable I used a string. After this all the data was ready to be added to an employee object. To do this I simply called the constructor and gave the parameters. Then I added the Employee object to the vector. We're using vectors because it's easier to print out all the objects in a vector using a for loop and *vector.size.()*. After all the data in the file has been parsed an added to a vector I can cout the list of employees stored in the vector using a for loop and my overloaded '<<' function for couting employee type objects.

Test Case

```
Program written by: Sheharyar Khan
Course info: CS - 116: Lab 2
Date: Tue Oct 16 21:56:09 2018

Name            ID      Dept            Position
----            --      ----            --------
Susan Meyers    47889   Accounting      Vice President
Mark Jones      39119   IT              Software Engineer
Joy Rogers      81774   Manufacturing   Engineer
```