```c
/*
 ===========================================================================
 Name        : assignment2.c
 Author      : Nathaniel Churchill
 Date        : 9/11/17
 Description : Prints a company report from the given files
 ===========================================================================
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "dept.h"
#include "emp.h"
#include "company.h"

Comp *company = NULL;

Emp *readEmployee(FILE *employeeFile);

Dept *readDept(FILE *fileName) {
        char deptCode[5];
        char deptName[20];
        char mgrEmpId[10];
        char buffer[1000];
        int numberReceived = fgets(buffer, 1000, fileName);
        if (numberReceived < 50) {
            return NULL;
        }

        int counter = 0;
        int wordCounter = 0;
        int offset =0;

        for (counter = 0; counter < strlen(buffer) + 1; counter++) {
            if (buffer[counter] == ',' || buffer[counter] == '\n' || counter == strlen(buffer)){
                if (wordCounter == 0) {
                    int numberBytes = counter - offset;
                    strncpy(deptCode, buffer, numberBytes);
                    deptCode[numberBytes] = '\0';
                    wordCounter++;
                    offset = counter;
                }else if (wordCounter == 1 ){
                    int numberBytes = counter - offset -1;
                    strncpy(deptName, buffer + offset +1, numberBytes);
                    deptName[numberBytes] = '\0';
                    wordCounter++;
                    offset = counter;
                }else {
                    int numberBytes = counter - offset -1;
                    strncpy(mgrEmpId, buffer + offset +1, numberBytes);
                    mgrEmpId[numberBytes] = '\0';
                    wordCounter++;
                    break;
                }
            }
        }

        //create the department
        Dept *createdDept = allocDept(deptName, deptCode, mgrEmpId);
        return createdDept;
}

/**
 * Load departments from a text file.
 * fileName : the filename of the file that contains the departments.
 */
void loadDepts(char *fileName) {
    FILE *departmentFile = fopen(fileName, "r");
    if(!fileName){
        printf("There was an error opening the department file\n");
    }
    Dept *department = NULL;

    while ((department = readDept(departmentFile)) != NULL) {
        addDepartment(company, department);
    }

    fclose(departmentFile);

}


/**
 * Load Employees from a binary file.  The employees are added to the list of employees
 * for their respective Department as indicated by deptCode.
 *
 * fileName : the name of the file to open
 */
void loadEmployees(char *fileName) {
    FILE *employeeFile = fopen(fileName, "rb");

    if(!employeeFile){
        printf("There was an error opening the employees file\n");
        return;
    }

    Emp * emp;

    while ((emp = readEmployee(employeeFile)) != NULL) {
        Dept * dept =  findDepartment(company, emp->deptCode);
        addEmployee(dept, emp);
    }


    fclose(employeeFile);
}


/*
 * Helper class to read employees from a binary employee file.
 * The function readEmployee() returns the next employee from
 * the employee file.
 *
 * @author Nathaniel Churchill
 */
Emp *readEmployee(FILE *employeeFile) {
    Emp *createdEmp = NULL;
    char buffer[51];
    int numberUsed = 0;
    int counter = 0;

    // Declare byte buffers for the ascii fields.
    char employeeIdBytes[4];
    char empIndicator[2];
    char deptCode[2];
    char lastNameBytes[21];
    char firstNameBytes[16];
    union _salaries {
        char csalary[4];
        float fSalary;
    } salary;

    int imonth = NULL;
    int iday = NULL;

    union _year {
        char cyear[2];
        short syear;
    } year;
    int iyear = NULL;

    union _vacationDays {
        char cdays[2];
        short sdays;
    } vacationDays;
    int ivacationDays = NULL;
    char training = NULL;


    int numberRead = fread(buffer, 1, 51, employeeFile);

    if (numberRead < 51) { //validate the number read
        return createdEmp;
    }

    //get the id
    for (counter = 0; counter < 3; counter++ ) {
        employeeIdBytes[counter] = buffer[numberUsed++];
    }
    employeeIdBytes[counter] = '\0';

    //get the emp Indicator
    empIndicator[0] = buffer[numberUsed++];
    empIndicator[1] = '\0';

    //get the deptCode
    deptCode[0] = buffer[numberUsed++];
    deptCode[1] = '\0';

    //get the last name
    for (counter = 0; counter < 20; counter++) {
        if (isalpha(buffer[numberUsed])){
            lastNameBytes[counter] = buffer[numberUsed++];
        }   else {
            lastNameBytes[counter +1] = '\0';
            numberUsed++;
        }
    }

    //get the first name
    for (counter = 0; counter < 15; counter++) {
        if (isalpha(buffer[numberUsed])){
            firstNameBytes[counter] = buffer[numberUsed++];
        }   else {
            firstNameBytes[counter] = '\0';
            numberUsed++;
        }
    }

    //get the salary
    for(counter = 3; counter >= 0; counter--) {
        salary.csalary[counter] = buffer[numberUsed++];
    }

    //get the month
    imonth = (int) buffer[numberUsed++];

    //get the day
    iday = (int) buffer[numberUsed++];

    //get the year
    for (counter = 1; counter >= 0; counter--){
        year.cyear[counter] = buffer[numberUsed++];
    }

    //get the year
    iyear =  (int) year.syear;

    //get the vacation days
    for (counter = 1; counter >= 0; counter--) {
        vacationDays.cdays[counter] = buffer[numberUsed++];
    }
    ivacationDays = (int) vacationDays.sdays;

    Date *date = allocDate(imonth, iday, iyear);

    //get  the training byte
    training = buffer[numberUsed++];

    //create an emp
    createdEmp  = allocEmp(employeeIdBytes, empIndicator, deptCode, firstNameBytes, lastNameBytes, salary.fSalary, date, ivacationDays, training);
    return createdEmp;
}



```

```c
/**
 * Prints a company report.  Report include information on the department
 * and a list of all employees.
 */
void printCompanyReport() {
    // loop over all departments
    int i = 0;
    for (i = 0; i < company->noDepts; i++) {
        Dept *department = company->departments[i];

        // print the department header
        printf("%s Department\n", department->deptName);
        getManager(department);//get the manager
        printf("%-20s%-10s%-10s\n", "Manager: ", department->manager->firstName,
department->manager->lastName);
        printf("%-20s%-20d\n", "Staff Size: ", department->noEmps);
        printf("%-20s%d\n", "Vacation Days: ", getTotalVacationDays(department));

        // print the column labels for employees
        printf("%-5s %-23s %-12s %-8s %-10s\n", "ID",
                "Employee Name", "Hire Date", "Salary", "Vac Days");

        // loop over all employees in the department
        for (int j = 0; j < department->noEmps; j++) {
            Emp *emp = company->departments[i]->emps[j];
            printf("%-5s %-11s %-11s %02d/%02d/%04d $%6.2f %6d\n", emp->empId, emp->firstName,
                    emp->lastName, emp->hireDate->month, emp->hireDate->day,
emp->hireDate->year,
                    emp->salary, emp->vacationDays);
        }

        printf("\n\n");
    }
}




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

    //allocate the company
    company = allocComp();

    if (argc < 2) {
        printf("Usage: companyReport <department file> <employee file>");
        exit(1);
    }

    loadDepts(argv[1]);
    loadEmployees(argv[2]);
    printCompanyReport();

    return EXIT_SUCCESS;
}

```