Lab 6 - section F

We'll assume the following employee record structure:

```
typedef struct {
    int empID, age, ssn;
    float salary;
} Record;
```

In this lab, you need to implement three functions based on the given structure definition above:

```
int readRecordFile(char *filename, Record ***array);
void freeRecordArray(Record ***array, int numElems);
void adjustSalaries(Record **array, int age, float x, int size);
```

The first two functions follow the same instruction as the prelab. The third function multiplies the salary of each employee whose age is equal to age by the value x.

Main program steps:

1. Create a Record pointer array by using your "readRecordFile" function. All the array element values should be initialized in your function as well.
Note: Each of the lines in the given csv file contains one employee's information in this order: salary, age, ID, SSN

2. Multiply the salary by 1.05 for the employee whose age is 30.

3. Free all the allocated memories by using your "freeRecordArray" function.

Grading Criteria:
Main program: 5 points
readRecordFile function: 20 points
freeRecordArray function: 10 points
adjustSalaries function: 5 points
All or nothing rule when grading: starting from lab 6, no partial credits will be given for incomplete/incorrect functions.

Note:

1. Uncomment the first line in starter code if you are going to use Visual Studio for this lab.
2. Please place “employee.csv” in the location where you have the source files. For example: Place “employee.csv” in the folder where you have the source.c file
3. To compile strictly with C flags, please follow the following instruction when you add your source file in VS: Add -> Add new item -> Source.c
4. Please make sure that your source file is in “.c” extension, NOT “.cpp” extension.

General note:

1. If your code does not compile, you will receive an automatic 0 for this assignment.
2. Changing the given function prototype or struct definition will lead to an automatic zero grade.
3. Using any global variables will lead to an automatic zero grade.
4. The implementation of the function should include comments describing what it is intended to do and how this function should be called. Example can be found in CS 2050 lab policy.
5. If your submission does not include a source file, you will receive an automatic zero grade.