Kadir Has University
Department of Computer Engineering
CE 241 – Programming Languages
Fall 2008 - Ahmet Ardal
Homework #9

1. Write a function that takes a file path string, an array of Grade objects and number of Grades as input. Inside the function, do the following:
- Create a file with name `filePath`,
- For each Grade in the array pointed by `pGrades`; write student id, a tab character(`'\t'`), grade and a new line(`endl`) to the file.

Function Prototype:
```
void gradeArrayToFile(const char *filePath, const Grade *pGrades, int nGrades);
```

2. Write a function that takes a file path string as input. Inside the function, do the following:
- Open the file named `filePath`,
- Until reaching the end of the file, read id-grade pairs from the file to a temporary Grade object,
- For each id-grade pair you read from the file into a Grade object, print that Grade object's id and grade members.

Function Prototype:
```
void printGradeFile(const char *filePath);
```

3. Write a function that takes a file path string as input. Inside the function, do the following:
- Open the file named `filePath`,
- Until reaching the end of the file, read id-grade pairs from the file to a temporary Grade object,
- For each id-grade pair you read from the file, increment a counter variable to obtain the number of grades in the file.

Function Prototype:
```
int countGradesInGradeFile(const char *filePath);
```

4. Write a function that takes two file path strings as input. Inside the function, do the following:
- Open the file named `filePath`,
- Obtain the number of Grade items file contains and create a dynamic Grade array of that size,
- Read id-grade pairs from the file into Grade array's items,
- Once you fetched grades from the file to memory, sort those grades by using the function `sortGradeArrById()`. (Note that, `sortGradeArrById()` function's code is provided in the test code...)
- Create a file with name `sortedFilePath`, write sorted array content to that file as you are asked to do in the 1st question, or directly use the function you wrote for 1st question.

Function Prototype:
```
void sortGradeFileById(const char *filePath, const char *sortedFilePath);
```

Notes:

- Grade structure:

```
struct Grade
{
   int studId;
   int grade;
};
```

- In this assignment's context, we use files to read and write data related to Grade structure. We do this by representing a single Grade object as a line in the file which consists of its id, a tab character, its grade and a new line. This information might be useful when writing the functions you are asked to implement in the preceding questions. For instance:

  3	70

  The above line may represent a Grade structure whose `studId` member's value is 3 and `grade` member's value is 70.

- Test code for the functions you are asked to implement in the preceding questions:

```
#include <iostream>
#include <fstream>

using namespace std;

// struct declaration
struct Grade
{
    int studId;
    int grade;
};

// function declarations
void gradeArrayToFile(const char *filePath, const Grade *pGrades, int nGrades);
void printGradeFile(const char *filePath);
int  countGradesInGradeFile(const char *filePath);
void sortGradeFileById(const char *filePath, const char *sortedFilePath);
void sortGradeArrById(Grade *pGrades, int nGrades);

int main()
{
    const int N_GRADES = 5;
    const char *gradeFilePath = "grades.txt";
    const char *sortedGradeFilePath = "grades-sorted.txt";
    Grade grades[N_GRADES] = {{5, 55}, {4, 44}, {3, 33}, {2, 22}, {1, 11}};

    // write grades to file
    gradeArrayToFile(gradeFilePath, grades, N_GRADES);

    cout << "original grade file:" << endl;
    printGradeFile(gradeFilePath);

    // count how many grades exist in the file
    cout << endl << "# of grades: " << countGradesInGradeFile(gradeFilePath) << endl;

    // sort grades and write them to another file
```

```
    sortGradeFileById(gradeFilePath, sortedGradeFilePath);

    cout << endl << "sorted grade file:" << endl;
    printGradeFile(sortedGradeFilePath);

    return 0;
}

// function definitions
void gradeArrayToFile(const char *filePath, const Grade *pGrades, int nGrades)
{
    // create file “filePath”
    // ...
    // write each grade to file
    // ...
    // close the file
    // ...
}

void printGradeFile(const char *filePath)
{
    // open the file “filePath”
    // ...
    // read Grades from the file and print ids and grades of them
    // ...
    // close the file
    // ...
}

int countGradesInGradeFile(const char *filePath)
{
    // open the file “filePath”
    // ...
    // read Grades from the file and count them
    // ...
    // close the file and return number of Grades
    // ...
}

void sortGradeFileById(const char *filePath, const char *sortedFilePath)
{
    // get number of grades the file named “filePath” contains
    // ...
    // create a dynamic Grade array of that size
    // ...
    // open the file “filePath”
    // ...
    // read Grades from the file and store each in the dynamic array
    // ...
    // close the file “filePath”
    // ...
    // sort grade array by id
    // ...
    // create a file to write sorted Grade array with name “sortedFilePath”
    // ...
    // write each grade to that file
    // ...
    // close the file and delete the dynamic array
    // ...
}
```

```
void sortGradeArrById(Grade *pGrades, int nGrades)
{
    Grade tmpGrade;

    // sort the grade array by id field using bubble sort sorting algorithm
    for (int i = 0; i < (nGrades - 1); ++i)
    {
        for (int k = 0; k < (nGrades - i - 1); ++k)
        {
            if (pGrades[k].studId > pGrades[k + 1].studId)
            {
                tmpGrade = pGrades[k];
                pGrades[k] = pGrades[k + 1];
                pGrades[k + 1] = tmpGrade;
            }
        }
    }
}
```

- Output of the test code for a sample execution:

```
original grade file:
student id: 5, grade: 55
student id: 4, grade: 44
student id: 3, grade: 33
student id: 2, grade: 22
student id: 1, grade: 11

# of grades: 5

sorted grade file:
student id: 1, grade: 11
student id: 2, grade: 22
student id: 3, grade: 33
student id: 4, grade: 44
student id: 5, grade: 55
Press any key to continue . . .
```

- After executing the test code above file contents look like this:

grades.txt:

```
5   55
4   44
3   33
2   22
1   11
```

grades-sorted.txt:

```
1   11
2   22
3   33
4   44
5   55
```