# *Assignment 1-part 2*

| | |
|---|---|
| ***Deadline:*** | **Anytime before** Sunday, 24th April 2016, 23:59 (midnight) |
| ***Evaluation:*** | 10 marks – which is 5% of your final grade |
| ***Late Submission:*** | 5% per hour (or fraction of hour) it is late |
| ***Teams:*** | The assignment can be done individually or in pairs (of at most 2 students) |
| ***Purpose:*** | Practice with C++ classes, encapsulation, data hiding, overloading. |

***Problem to solve (same problem as in Assignment 1-part 1, but using classes, encapsulation, data hiding and not structures):***

The University registrar maintains a number of computer databases: student information, courses offered, enrollments, etc. One of the problems with maintaining all this information is keeping it accurate. In time, errors will creep into the University's databases unless programs are run from time to time to check for errors. In this assignment you'll write such a program.

In short, your program will read from two text files, "**students.txt**" and "**courses.txt**", and output any inconsistencies it discovers. Name your program ***a1p2.cpp***

**Problem Statement:** Write a program that does the following:

- Displays on the standard output information about the authors of the program (IDs , family & given names, , assignment number)
- Reads all data contained in the files "students.txt" and "courses.txt";
- Writes out, on the standard output, any inconsistencies it discovers. Your program should perform the following three consistency checks:
  1. Are there any students not enrolled in any course?
  2. Are there any courses with no students enrolled?
  3. Are there any courses with enrolled students that do not exist?

In each case, each student or paper that yields a "yes" answer to the question should be output on the screen. To help with testing, an example of the input files is provided below.

**Figure 1**

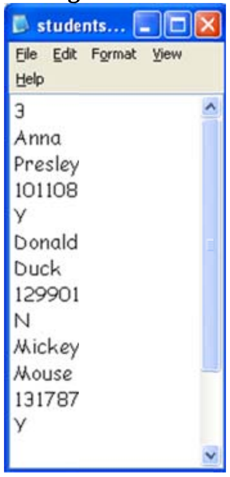

```
3
Anna
Presley
101108
Y
Donald
Duck
129901
N
Mickey
Mouse
131787
Y
```

**Figure 2**

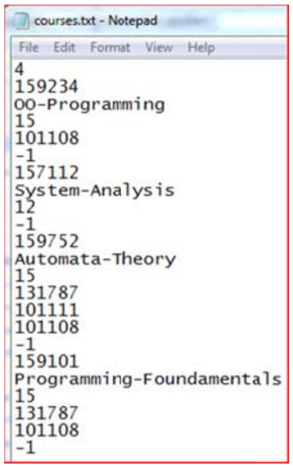
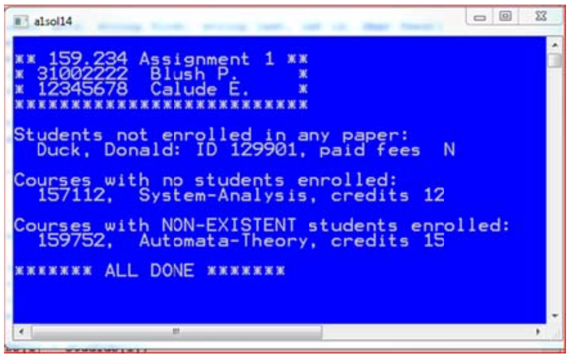

```
4
159234
OO-Programming
15
101108
-1
157112
System-Analysis
12
-1
159752
Automata-Theory
15
131787
101111
101108
-1
159101
Programming-Foundamentals
15
131787
101108
-1
```

**Figure 3**


```
** 159.234 Assignment 1 **
* 31002222  Blush P.    *
* 12345678  Calude E.   *
*************************

Students not enrolled in any paper:
   Duck, Donald: ID 129901, paid fees  N

Courses with no students enrolled:
   157112,  System-Analysis, credits 12

Courses with NON-EXISTENT students enrolled:
   159752,  Automata-Theory, credits 15

******* ALL DONE *******
```

The format of the "*students.txt*" file is as follows. It's a file with at most 100 students, where the following information is stored: an integer representing the total number of students and for each student (in this order): first name (one word), last name (one word), id number, and Y or N depending on whether the student has paid the fees or not.. The students are listed in ascending order by student id, and each piece of information is stored on a line by itself. See Figure 1 for an example.

The "*courses.txt*" file contains information about at most 50 courses, with the following information: an

integer representing the total number of courses offered and for each course: the course identifier, the course title (one word), number of credits, and a list of 0 or more student ids followed by –1. See Figure 2 for an example.
Given the sample input files shown in Figure 1 and Figure 2, your program should produce an output as shown in Figure 3.
Note that you are required to match this output format as closely as possible, to enable uniformity in the grading process.

In solving this assignment you should comply with the following restrictions.

a) Your solution must contain at least two `classes`, one called **`Student`** that represents a single student, and another called **`Course`** that represents a single course.
b) You have to read the information from the input files into some sort of data structure (I recommend an array of Student(s) and an array of Course(s)); you are not allowed to repeatedly read from the input files over and over in order to perform the consistency checks.
c) Your program should open the files *students.txt* and *courses.txt* without asking the user to input the names of the files from the keyboard.
d) Your program should be organized as follows: header comments first, all included files, class Student listing, class Course listing, class Student implementation, class Course implementation, function prototypes, main function followed by all function definitions.
e) Do not use dynamic memory allocation (no linked lists,…) for this assignment ; marks will be deducted in case you do use these features.
**f) Important. All data members in the classes you design in this assignment should be private and access to them should be provided using public member functions/methods only. If a class has one or more public data members the mark for the whole a1p2 will be 0 marks.**

**Hand-in**:
Submit **a1p2.cpp** electronically using STREAM.

***Miscellaneous:***
1. Marks will be allocated for: correctness, completeness, use of C++ constructs presented in class/labs, simple and clear solution, good documentation, and structured output display (on the screen).
2. Using goto, **global variables** or C-like I/O constructs (i.e *printf, fprintf, scanf, FILE**, etc) is not allowed and it will be penalised. Only const global variables are allowed.
**3.** Programs that **do not run** **or** **do not compile in the (Albany) labs, using gcc(SciTe), get 0 marks.**
4. Suspicious **similar solutions** will **all** get 0 marks.
5. Write YOUR ID NUMBER(S), and YOUR **FAMILY** NAME(S) first, assignment number, what the program does at the beginning of the file you send electronically and *at least* comment each function.
6. When working in teams, **send one solution file per team.**

**If you have any questions about this assignment, please ask the lecturer before its due time!**