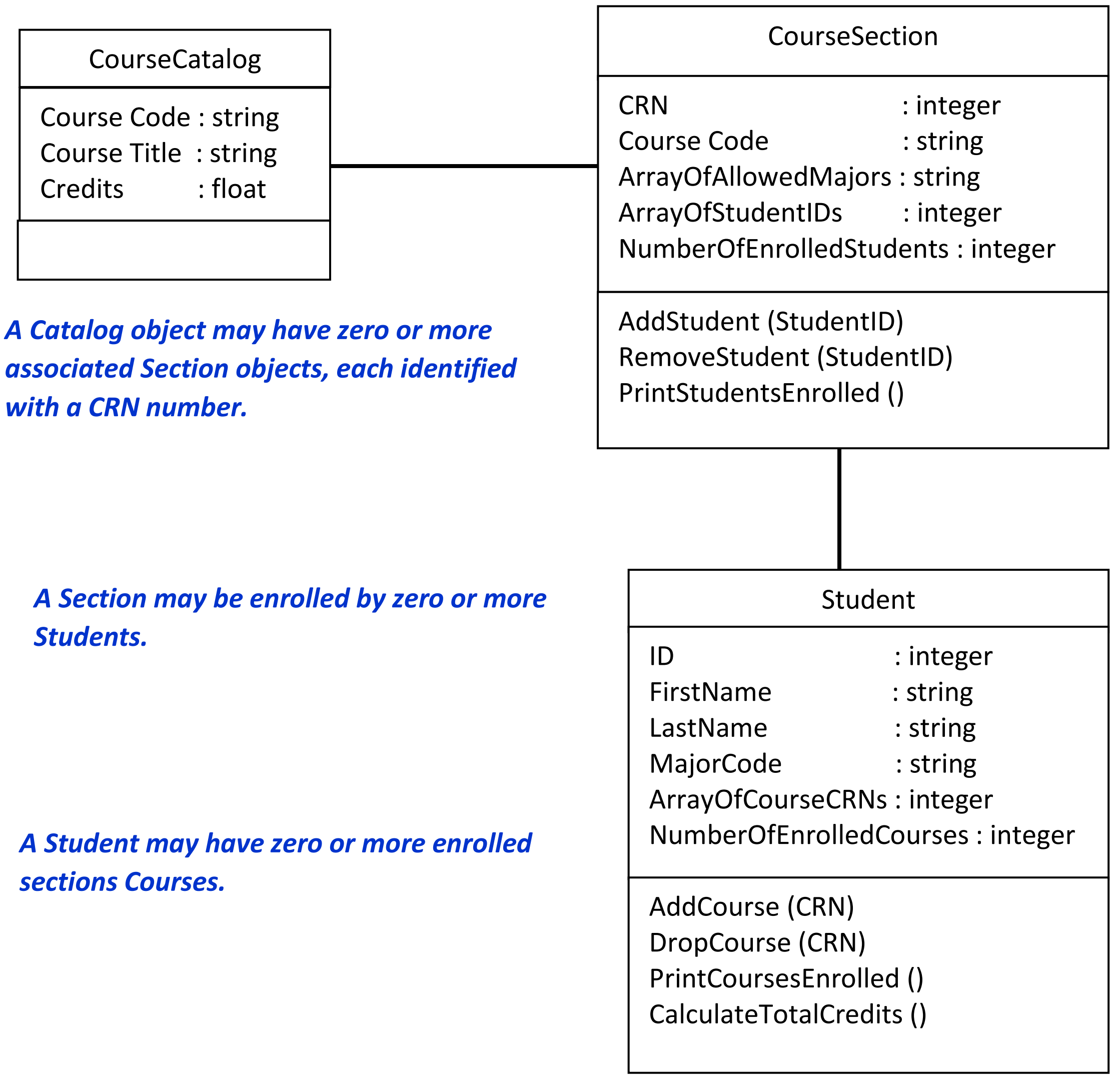
BLG252E - Object Oriented Programming

HOMEWORK #1

Due Date: 24.10.2013

The following diagram shows three C++ classes for a school.

Upper sections are class names, middle sections are attributes (member data), and lower sections are methods (functions).



A Catalog object may have zero or more associated Section objects, each identified with a CRN number.

A Section may be enrolled by zero or more Students.

A Student may have zero or more enrolled sections Courses.

Write a C++ program to do the followings:

1) Read the input text data files (CATALOG.TXT, SECTIONS.TXT, and STUDENTS.TXT) and initialize your objects. (For file reading, you may use either the C functions fscanf and fgets, or the C++ object ifstream).

2) Read the input transactions file (TRANSACTIONS.TXT) which contains several transaction directives (as uppercase strings) and parameter data, and perform the requested actions.

There are 5 possible transaction directives (i.e., operations) which are described below.

Transaction Directive	Transaction Parameters	Things to do
ENROLL	<CRN> <List of Student ID numbers>	Add the specified students to the given course section. (You should consider it as a sequence of several ADD actions.)
ADD	<CRN> <Student ID number>	Add the student to the given course section.
DROP	<CRN> <Student ID number>	Remove the student from the given course section.
PRINT_SECTION	<CRN>	Print list of students (ID, First name, Last name, students' MajorCode) that are enrolled to the given course section. Also total number of students enrolled should be displayed.
PRINT_ STUDENT	< Student ID number >	Print list of course sections (CRN, Course Code, Course Title, Credits) that the student is enrolled. Also total credits of the student should be calculated and displayed.

SAMPLE INPUT DATA FILES (4 Files):

CATALOG.TXT

```
MAT101 Mathematics 4
MAT105 Calculus 4
ELE101 Electronics 3
FIZ101 Physics 2
……
```

STUDENTS.TXT

```
10030066 Abdullah  Akkaya MET
10050183 Adnan Erguven    TEK
10070045 Ahmet Pekkoc     ISL
10070090 Ahmet Avci       ISL
10070046 Ahmet Ozdemir    INS
10080049 Ahmet Ulas       GMI
10080094 Asli Karacayli   KIM
……
……
……
```

SECTIONS.TXT

```
6001  MAT105  JEO MAD INS
6002  MAT105  JEO MAD INS
7001  MAT101  BIO GID GMI INS ISL JEO KIM KMM MAD MET TEK
7002  MAT101  BIO GID GMI INS ISL JEO KIM KMM MAD MET TEK
7003  MAT101  BIO GID GMI INS ISL JEO KIM KMM MAD MET TEK
8001  ELE101  GMI TEK
8002  ELE101  GMI TEK
9001  FIZ101  INS
……
……
```

TRANSACTIONS.TXT

```
ENROLL   6001   130100204  70080016  140080024
ENROLL   6002   130080015  60100119  60100419  90100335  60080179
ENROLL   7001   70100025  130100204  60100119  140100028  10030066  130100272  70080016
ENROLL   7002   70100074  70100069  90090270  70110128  10100136  10080125  10110065
ENROLL   8001   70080016  10080097  70100025  140090046  10050183  130120701
ENROLL   8002   70110050  60060124  60110810  10030066  70080016
ENROLL   9001   60100119  70080016  50080129  60080155  70100025  10100553  70080023
ADD  7001  60100119
ADD  7001  140080016
ADD  7002  70090072
DROP  7001  130100272
DROP  7002  90090270
PRINT_SECTION  6001
PRINT_SECTION  7001
PRINT_SECTION  7002
PRINT_SECTION  8002
PRINT_STUDENT  70100025
PRINT_STUDENT  70080016
……
……
```

OTHER REQUIREMENTS:

- When adding a student to a course section, your program should check the student's major code with the allowed major codes for that section. Display a warning message if the ADD operation fails.

- The TotalCredits of a student can not be more than 10 credits. Display a warning message if the ADD operation fails for that reason.

- Write appropriate class constructors and destructors in your code.

- Your program should be general, and should be independent of the sample data given in the input files.

In order to tokenize (i.e., splitting into separate words) a sentence of unknown size, you may use the function **`tokenlara_ayir()`** given below.

```
#include <iostream>
#include <string.h>
using namespace std;
#define N 20    // Maximum number of words
#define LEN 15  // Maximum length of a word

void tokenlara_ayir( char * cumle, char  dizi[N][LEN])
// Input : cumle
// Output : dizi
{
   int i=0;
   char *tokenPtr; // Define word pointer

   tokenPtr = strtok( cumle, " " ); // Begin tokenizing sentence.

   // Continue tokenizing sentence until tokenPtr becomes NULL.
   while ( tokenPtr != NULL ) {
      strcpy(dizi[i] , tokenPtr );
      i++;
      tokenPtr = strtok( NULL, " " ); // Get next token
   }
}

int main() {
    char ornek_cumle[200] = "aaaa bbbbb cccc dddd";
    char kelimeler[N][LEN];
    int i;
    for (i=0; i< N; i++)
          strcpy(kelimeler[i] , "");

    tokenlara_ayir(ornek_cumle, kelimeler);
    for (i=0; i< N; i++)
      if (strcmp(kelimeler[i], "") != 0 )
               cout << i+1 << " = " << kelimeler[i]<< endl;
   return 0;
}
```