Computer Programming
Final Exam
Spring 2012

National University of Computer and Emerging Sciences

Marks: 100 **Time:** 3 hrs.

Q1. This problem tests your knowledge of classes, composition, inheritance, polymorphism, file handling and exception handling. Read the questions carefully and answer precisely. [**Marks:** 10 + 15 + 10 = 35]

Note: If you're not familiar with the C++ string you can use the simple char* in the following definitions.

(a) Define and implement a class called TableOfContents. It contains a dynamic array of type Content, defined by the following struct:

```
struct Content{
        string name;//chapter  name
        int pno;//page number of the chapter
};
```

TableOfContents also contains an integer n for the number of contents. The class also has a constructor **TableOfContents(const string & fname)** which receives a file name and reads the contents from that file. It is a text file in which the first line contains the number of chapters and then alternate lines contain chapter name and corresponding page number. Following is an example file with only two contents.

2
First Content
1
Second Content
10

If the file cannot be opened your program should throw an exception and catch and report it appropriately. The class should implement a function called **printContents** which prints chapter names and the corresponding page numbers, line by line.

(b) Read the following description carefully and implement a system of classes accordingly. For each class you must implement appropriate constructors and methods only. For the Date class you can simply use the following interface without implementing it.

```
class Date{
      int day,month,year;
public:
      Date(int,int,int);
      void SetDate(int,int,int);
      int operator – (const Date&);
      //difference between two dates in number of days
      };
```

A Book *contains* a table of contents, a string ISBN (which is unique for each book) and a string title. A reference book , which cannot be issued, *is a* book, and an issuable book *is* also a book. All books can print their table of contents but reference books print only the number of chapters and page numbers of the first and the last chapter, while issuable books actually print all the chapter name and their corresponding page numbers, line by line.

A ***dictionary*** and an ***encyclopedia*** are both reference books, while a ***text book*** and a ***general book*** are both issuable books. Every reference book can print a short description of itself. The description of a dictionary is: "**title** is a list of words and their meanings", and the description for an encyclopedia is: "**title** contains articles on various topics", where **title** is simply the title of the book. Every issuable book contains a date of issuance and can calculate fine for a book by checking if the return date (passed as parameter) is more than fourteen days away (in case of general book) or more than seven days away (in case of text book) from the date of issuance. A 10 rupee fine per day is charged for a text book and a 5 rupee fine per day is charged for a general book. An issuable book stores the last value of calculated fine (which is set to zero in the constructor), for later use. Issuable books can be issued by setting the issuance date to a date passed as parameter. The system only allows objects of the four types: ***dictionary, encyclopedia, general book, and text book,*** to be instantiated.

(c) Implement a class called Library which contains dynamic arrays of issuable and reference books, called shelf1 and shelf2 respectively, and the numbers of items in both. For this class you must implement the following methods (Note: You might assume that all the data is already stored in the library; you don't need to take any input):

AverageLastFineForIssuabeBooks: this function goes through the array shelf1 and calculates the average fine (over the values of last calculated fine stored in each issuable book). Average fine is simply the total last fine divided by the total number of issuable books. *In this function you must properly handle the divide by zero exception.*
printBookDescriptions: this function goes through the array shelf2 and prints the descriptions of all the reference books inside it.
printBookDetail: this function receives a pointer to a single book of any kind, and prints its ISBN, title, and table of contents.

Q2. [Marks 10*2=20] Answer the following questions briefly:

i) Write only the prototype of the method in class A which will enable the line: `A obj = 3.45;` to execute without any error or warning.
ii) We want to specialize a class template called Vector for the type **bool**. What will be the syntax for doing that? (You don't need to define the class vector. Just give the syntax for template specialization).
iii) What are three benefits of using inheritance/polymorphism?
iv) What is the difference between private and protected keywords?
v) Correct the following code and explain your corrections.

```
Class B{
   int x;
public:
   B(int i){x=i;}
};

class A{
   B b;
   const int y;
public:
   A(int j, int k){y=j; b.x=k;}
};
```

vi) Suppose we are implementing class A which will make use of all the methods and data of class B. We can accomplish this by either inheriting from class B or making class A contain an object of class B. What should be our criteria for choosing any one of these methods?

vii) If a global function wants to access the private data members of a class, it can do so using the getter functions of the class. What is the other method for allowing this access? What are the relative advantages and disadvantages of these two methods?

viii) For the following code, add appropriate catch blocks so that when the index is less than zero you print "Index too small", and when it is greater than 4 you print, "Index too large".

```
Int square[5]={0,1,4,9,16},index;
try{
    cout<<"Enter number between 0 and 4";
    cin>>index;
    if(index > 4)
          throw index;
    else if(index < 0 ) throw false;
    else cout<<square[index]<<endl;
   }
```

ix) While overloading an operator like +, -, * and /, which returns a new object as answer, the result is returned by value. However, while overloading the assignment operator the result is retuned as reference. Explain this difference, justifying the return type in each case.

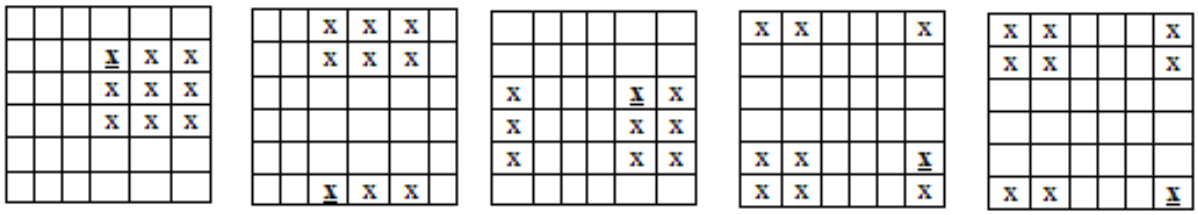
x) Following is a C++ class representing a mathematical fraction, where *n* is the numerator and *d* is the denominator. Implement the post- and pre- increment operators for this class.

```
Class Fraction{
         int n,d;
   public:
};
```

Q3. [Marks 20] Write a C++ function called findMatrix which searches for a square matrix inside another square matrix. A square matrix is one in which the number of rows is the same as the number of columns. Your function should accept the two matrices and their dimensions as parameters, and return true, if the second matrix is found inside the first one, and false otherwise. Note: it is possible that the second matrix

occurs inside the first matrix after wrapping around horizontally or vertically, as shown in the following two pictures. The x's denote a possible region of the bigger matrix where a smaller one may occur. The x on the top-left corner of a smaller matrix is bold and underlined. In these examples the dimensions of the matrices are 6x6 and 3x3 respectively. Please note: these are only examples. Your function should deal with all possible cases for all possible dimensions, not just these ones.

Examples:



Q4. [Marks 25] Implement a C++ class called Array which satisfies all the requirements of the following main program. Array contains a dynamic array of integers. Following is a partial definition of this class:

```
class Array{
      int * a;//dynamic array
      int cap;//capacity (total space in the array)
      int n;  //size (number of elements inserted)
 public:
      //…};
```

Description for the functions is given in comments below calls in the following program; read them carefully. Also, you implement all the necessary constructors/destructors and operators used in the following:

```
int main()
{
      int temp[]={12,4,6,11,5};
      Array A;
      /*make an empty array of capacity 0*/

      Array B(temp,5,10);
      /*Make an array with size 5 and capacity 10. If no capacity is
      specified or capacity is less than size, then set capacity equal to
      size, and allocate the array of that capacity.*/

      int n, input;
      cin>>n;
      for(int i=0;i<n;i++){
            cin>>input;
            A.insert(i*2);
            /*
               Insert puts the new integer at the next available index
               in the array. If size exceeds capaity then, a new array
               of double capacity is allocated, data is copied into it
               and the older array is de-allocated.
            */
      }
```

```
        A.printAndDeleteRange(5,15);
        /*
        All elements in the given range are printed and then removed      from
        the array. The size of the array is reduced accordingly. If the new size
        is less than half of the capacity, the capacity is halved; a new array
        of half size is allocated. Data is copied into it    and old array is
        deleted.*/

        A.reverseInRange(3,7);
        /*
        The order of all elements in the given range is reversed,    keeping the
        remaining elements un-disturbed. For example, if the array
        contained 2,6,8,4,1,13,7,5, then after this call it will become:
        2,5,8,7,1,13,4,6 (notice the elements in bold have been reversed)
        */

         Array C=A-B;//All elements in B are removed from A
         A=C;

         cout<<C;//print the array in C

         return 0;
}
```

THE END