CSCD-305, C++ Programming
HW 2
Due: 2/20/2012

We will begin developing a set of classes connected by inheritance, one of possible utility for a ticket sales agency that sells a number of different kinds of tickets. The initial exercise will be to set up a *few* kinds of tickets. The hierarchy will then be expanded in the next assignment. We will also take advantage of code *reuse*: we will use the Deitel book code as a small class library and use the Date and Time classes developed there.

The base class will be a generic ticket. Attributes of a generic ticket will be as follows:

- Date of the performance — using the class Date, available in Date1.h and Date1.cpp through
- Time of the performance — using the class Time, available in Time3.h and Time3.cpp through.

Amplify it by adding `ostream &operator << (ostream &Out, const Time &T);`
You can copy the code from `Time::printStandard()`. Replace "`cout`" with "`Out`"; replace data member accesses (such as `hour`) with appropriate get functions (such as `T.getHour()`). Only print the hour and the minute. Then end with the "`return Out;`". This way the operator << doesn't need to be a friend function to Time and you don't have to change the Time3.h file.

- Location of the performance — using a dynamically allocated character string
- The unique ticket number (as a `const`), based on a static class member "`m_NextNumber`".

Behavior associated with a generic ticket:

- Read in the above information, and set the `const int m_TicketNumber` from the data member `static int m_NextNumber`. This is within the constructor. If you wish, though, you may write a "Set" function that is called by your constructor.
- Print the above information. Make this "`void Print(ostream&);`" as a public member function.

Derived classes are the following, shown with their attributes:

- Popular music concert
    - ♦ List of groups performing — using a dynamic array of dynamically allocated character strings.
- Classical music concert. Projected later developments in these classes will be facilitated by having an explicit base class for classical music, from which we will derive various types of classical concerts.
    - ♦ Small ensemble concert (for instance, string quartet)
        - ❑ Name of the ensemble — using a dynamically allocated character string.
    - ♦ Solo recital
        - ❑ Name of the soloist — using a dynamically allocated character string.
        - ❑ Name of the accompanist — using a dynamically allocated character string, possibly a NULL pointer, if "accompanist" is inappropriate (as for a solo piano recital or some other musical form that is performed unaccompanied).

Behavior associated with each of the above derived classes (except for the generic classical ticket class):

- Read in the appropriate information. This is within the constructor.
- Print the appropriate information (after calling `CTicket::Print`). Make this a public member function — "`void Print(ostream&);`".

Have your constructors expect to receive an istream — reading information from a file. The *very first* line of the input file will be the value to be used for initializing the `m_NextNumber` static data member, and will actually be consumed by the `main` (in this implementation). Then in the `main` the logic for generating tickets will itself consume the first line of a ticket record to determine which kind of a ticket to generate. It will use the ">>" operator, and so will skip past white-space automatically.

Put debugging code into all of your constructors and destructors to report to `cout` the entry into that function. The code should be enabled by putting at the top of your implementation file the declaration `#define DEBUG`", and disabled by commenting that line out. In other words, the debug sections in the constructors and destructors will begin with the line "`#ifdef DEBUG`" and end with the line "`#endif`".

Input format for Ticket records:

- First line: character string indicating ticket type — "Pops", "Ensemble", or "Recital". The `main` provided uses the `<string.h>` function `stricmp` to perform a case-***in***sensitive comparison. If the record has any other value, the program reports an error and terminates processing.
- Second line: date and time, written in the form "mm/dd/yyyy hh:mm" Note that the time in the file will be for the 24-hour clock (that is, in the form expected by the Time constructor).
- Third line: location, with possible embedded blanks (i.e., use `getline` rather than `>>`).
- Subsequent lines: as required by the ticket type.
  - ♦ The "Pops" ticket type will have first the number of acts on the bill, followed by as many lines as are required to give those acts.
  - ♦ The "Ensemble" ticket type will have a single line giving the ensemble information
  - ♦ The "Recital" ticket type will have soloist information as the fourth line, and accompanist information as the fifth line. If the fifth line is completely blank, do *not* allocate space, but instead set the Accompanist pointer to NULL.

You will be given the main program to exercise your class implementations. Consequently it is necessary to specify the interface.

- Within CTicket: `static void SetNumber(int N) { m_NextNumber = N; }`
  The `main`, after it opens the input file, will read in the starting value for `m_NextNumber` and will execute the statement "`CTicket::SetNumber(NumIn);`".
- As mentioned above, you will have constructors that expect to receive a single parameter — `(istream&)`. Also, since all but one of the classes use dynamic memory, you must have explicit destructors. Indeed, the specifications regarding debugging code require you to have explicit destructors for all classes.
- In all classes, have a member function `void Print(ostream&);` — this is to send the formatted ticket to the ostream object passed as the parameter. The derived classes will begin their implementation of `Print` with a call to be the base class `Print`. In other words, the statement "`CTicket::Print(Out);`" will begin the `Print` function in all of the derived classes.

Here are sketches of the four possible ticket formats — popular music concert, small ensemble, solo recital with accompanist, and solo recital without accompanist. (You are not expected to generate a surrounding box in your print-out, since the behaviors of the Time and Date classes make it difficult to know how long an output line is when it includes a Time and/or a Date. It is *doable*, using a stream class called "strstream", but using that would require modification of the Time and Date classes.)

Popular music concert

```
Ticket No. 12345
7:30 pm  March 3, 2000
Showalter Auditorium

Presenting:
   Smashing Pumpkins
   Hurling Squash
```

Classical Music: small ensemble

```
Ticket No. 12346
8:30 pm  March 9, 2000
Showalter Auditorium

The EWU CSM&T String Quartet
```

Classical Music: solo recital without accompanist

```
Ticket No. 12347
7:30 pm  March 12, 2000
Showalter Auditorium

Recital by Samuel Smith, Marimba
```

Classical Music: solo recital with accompanist

```
Ticket No. 12348
7:00 pm  March 17, 2000
Showalter Auditorium

Recital by Marvin Minski, Counter Tenor
   Accompanied by Fingers Fortissimo, Piano
```

Data file that would generate the above — with a blank line following each ticket record to aid in readability.

```
12345
Pops
3/3/2000 19:30
Showalter Auditorium
2
Smashing Pumpkins
Hurling Squash

Ensemble
3/9/2000 20:30
Showalter Auditorium
The EWU CSM&T String Quartet
```

```
Recital
3/12/2000 1900
Showalter Auditorium
Samuel Smith, Marimba


Recital
3/17/2000 1900
Showalter Auditorium
Marvin Minski, Counter Tenor
Fingers Fortissimo, Piano
```

Below is a copy of the screen generated by running the main program on the above data file when the CTicket functions are compiled with `#define DEBUG` enabled.

```
 Input file name: Asg5Inp.txt
Output file name: Asg5Out.txt
CTicket constructor: March 3, 2000 7:30 PM
CPops constructor: Smashing Pumpkins
CPops DESTRUCTOR:  Smashing Pumpkins
CTicket DESTRUCTOR:  March 3, 2000 7:30 PM
CTicket constructor: March 9, 2000 8:30 PM
CClassical constructor:
CEnsemble constructor: The EWU CSM&T String Quartet
CEnsemble DESTRUCTOR:  The EWU CSM&T String Quartet
CClassical DESTRUCTOR:
CTicket DESTRUCTOR:  March 9, 2000 8:30 PM
CTicket constructor: March 12, 2000 7:00 PM
CClassical constructor:
CRecital constructor: Samuel Smith, Marimba
CRecital DESTRUCTOR:  Samuel Smith, Marimba
CClassical DESTRUCTOR:
CTicket DESTRUCTOR:  March 12, 2000 7:00 PM
CTicket constructor: March 17, 2000 7:00 PM
CClassical constructor:
CRecital constructor: Marvin Minski, Counter Tenor
CRecital DESTRUCTOR:  Marvin Minski, Counter Tenor
CClassical DESTRUCTOR:
CTicket DESTRUCTOR:  March 17, 2000 7:00 PM
```

`When you hand in the project, include a screen dump of a run with #define DEBUG enabled.`