**Project Cover Page**

Due date: April 29, 2013 Submission date: April 29, 2013

First Name: Last Name:

Alan Achtenberg

Jacob Stone

Section Number: 501

Homework Type & #: Project 2

Please list all sources in the table below: web pages, which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, see the link to student Rules

<http://student-rules.tamu.edu/aggiecode>,

|  |  |  |  |
| --- | --- | --- | --- |
| Type of sources |  |  |  |
| People |  |  |  |
| Web pages (provide URL) |  |  |  |
| Printed material |  |  |  |
| Other Sources |  |  |  |

I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work.

“On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.”

Your signature Typed Name Date

Report:

1. **Program Description**
   1. *Our program is designed to read in a list of purchased tickets from an input file, named “tickets-input.dat”, collect the information from the list, and then rewrite the list to a file named “tickets-output.dat”.*
2. **Purpose of the Assignment**
   1. *The purpose of this assignment was to recall everything we have learned over the semester and implement it into one single program. We were to work with classes, input/output files, functions, pointers, overloading operators and much more.*
3. **Data Structures**
   1. *In this program we have reinforced and gained knowledge in structures being pointed by its own type of pointers.*
   2. *We implemented pointers to our base class Tickets, and then we also made use of these pointers by using the arrow operator (->). We also implemented data structure in our calculate\_time function, in which we created two tm structs and then used them to compare time and dates.*
4. **Algorithm Description**
   1. *Besides simple functions created by us we did not make use of any algorithms (those actually defined in the Algorithm library).*
5. **Program Organization and Description of Classes**
   1. *We broke our code into 6 different files. Each file contains code that relates to everything else in the file. For example, our “ticket.h” contains the base class Ticket and its three subclasses: advance, student and walk. Another couple of example is our “functions.h” file that contains all of our functions for these classes, and our “our\_time.h” file that contains all of the functions necessary for date/time calculations.*
6. **C++ Object Oriented Features**
   1. *In our program we used inheritance, polymorphism and encapsulation.*
   2. *What made the ticket class abstract was that it was the base class of the three subclasses and it contained a pure virtual function that was implemented by the subclasses.*
   3. *Class Ticket was our abstract class and it held all the basic member functions of the following subclasses: advance, student and walk. Ticket had a pure virtual function print() and each of the subclasses override for their own use.*
7. **Exception Classes**
   1. *We did not use any exception classes in our code because the way our code is implemented, with check\_input functions and such. There was no need for the exceptions because our functions took care of the problems and would loop until the problems were resolved.*
8. **How to Compile and Run Our Program**
   1. *For ease of compilation we simply put all functions and classes into header files that way all you have to compile is the actual project\_main.cpp and you type a.out to run the program.*
9. **Input Specifications**
   1. *For the program to correctly input, on individual lines and in correct order the information must be provided with a : before for example Name:Alan would read in Alan not Name:Alan. The order of information is as follows, (Name, Date, Time, Purchase Date, Ticket number, Student info{yes or no})*
10. **Exceptions**
    1. *In the case that the input file is not found or opened correctly, our program throws an integer which is then caught and the corresponding error message is displayed. The program exits returning 2. In any other exception case possibly from something in std\_lib\_facilities\_3.h the message an exception has occurred is displayed and the program exits returning 1.*
11. **Testing**
    1. *For the valid cases walk up ticket Advance ticket(10 days before and less then 10) Student ticket(10 days before and less then 10)everything works correctly, for the invalid cases such as the purchase date being after the event date, everything still runs but the ticket price will be considered by the program to be a more than 10 days before ticket, it will mostly be undetected, for random incorrect input types the program can crash unexpectedly because of range errors or it can just display garbage values.*