Name: Jingtao Cheng

PSU ID: 940067494

Date: Mar. 2. 2018

CS202-Win-2018-Prog3

CS202 Program 3 Analysis and Debuggers

Analysis:

In this project, I was aiming to create an object-oriented program to help people watch their exciting events of 2017 Winter Olympics. This application can help the user find out when a particular sport will be taking place and even when we expect a specific athlete to be competing.

Design:

In this program, I have developed at least five classes to implement all the functions needed. They are Client, A\_node(node of athletes), B\_node(node of BST(balance tree-2 3 4 tree)), H\_node(node of historical records) and Sport class. First, the client class is used to manage all the main ability of this application and will be used in the main function by the user. So, the client class will contain the root node of the balance tree. The B\_node class is the central node of the balance tree which is also a derived class from the Sports class. It has left, and write pointers point to itself. It also has a + and += operator which can act as the copy constructor and insert function, the == operator can be used to compare. The sports class has four private string character which stores name, detail, time, channel. It also contains the head of the athlete's list. Besides, the class contains the copy constructor which can call the base class's constructor from the initialization list. And the go\_left, go\_right, set\_left, set\_right function which can get the left and right address and set the passed in reference to the left or right address. The A\_node class stores the information about each athlete. It has the name, ID; the history records list which is the head pointer point to a H\_node list. It also has go\_next, set\_next function and copies constructor to handle the dynamic memory. The class has the [] operator which can be used to search the athletes and +, the += operator which can be used to add In a new athlete or a new history record. The H\_node stores each historical records, which including the time and the event and the medal's information. It also has a copy constructor to handle the dynamic memory. In this program, I have not use statically allocated arrays in my classes or structures, and I used dynamically allocated arrays instead. I have put all my data member in the private section and use public and private functions to access them. I have not put the input operation in my class. Instead, I have put all the input operation in the client program, and I have used several .cpp file to implement the functions and use .h fine to define the functions, finally, using main.cpp to perform the all the client operation. I have not used global variables in my program, if necessary I have to define const constant. I have not use the string either string class, and I realized that I could use the string library to use several functions about string, like strlen(), strcmp(), and strcpy(). Every characters and sentence will be stored in the arrays of characters instead.

UNIX Debuggers:

In this program, there are many places have to use dynamically allocated memory including several pointers and dynamically allocated array to build the Doubly linked list and the linear linked list and all the other dynamically allocated array for the characters to form the whole data structure. So how to manage all the pointers correctly and invoke and modify them properly as well as efficient deallocate those pointers is a tough point. Using the gdb, I can get to know every pointer? Address and check the address with the arguments which I pass into the function. Also, I can access that pointer to see if the value was passed in or not to find and figure out the problem. I always use Valgrind to check my memory leak. (Valgrind ?tool =memcheck ?leak-check=yes ./a.out). During the debugging period, I've found many mistakes I made, such as Invalid write, which stand for I was using a pointer allocated and then use that pointer outside the range, and another reference fault, segmentation fault like invoke head -> function while the head was NULL and many other mistakes. Using the breakpoints, I can run my program until the breakpoint to a minimum the scope to find the specific problem.