Debugging with ddd

Today, we are going to look at how to debug programs with a debugger. We will be using the free debugger called "**ddd**" (The "Data Display Debugger").

Debuggers enable a programmer to see their program run in an environment that gives a tremendous amount of information about the program itself, and this is helpful for finding where the problems are in order to fix them.

We are going to debug the programs that we debugged by hand in Lab 5, and hopefully you will see that the process leads to fast debugging. There is a certain learning curve necessary for learning how to use a debugger, but in the long run it can be very beneficial.

Low Level Details

Getting the files:

At the command prompt, enter

```
cp /comp/15/public_html/labs/lab7/code/* .
```

Compiling and running:

1. At the command prompt, enter "**`make student=X`**" (where X is a number between 1 and 7) and press enter or return to compile.
2. At the command prompt, enter "**`./studentXcalendar`**" and press enter or return to run the new program.

Providing:

At the command prompt, enter "**`make provide"`** and press enter or return. For this lab, you will need to provide a list of bugs from **program listing 4**, and how you fixed them.

Please follow the instructions below in order to debug program 1. Once you complete program 1, go on and try to debug program 2 yourself. You can go on to the other programs, but again, please ensure you make it to debugging program listing 4 in order to provide your results.

1. First, copy the files into a directory (see above).

2. Type:
```
make student=1
```

3. Run program 1 with the following command, and note the errors:
```
./student1Calendar
```

4. Open ddd (`ddd&`) (*note:* you might want to do this from another terminal window).
5. `File->Open Program->student1Calendar`

5. Display line numbers (`Edit->preferences->Source tab->"Display Source Line Numbers"`)
6. The test for "add event on to end of day" failed, and was the first bug, so we will pursue this bug.
7. Put a breakpoint in that test (double-click at the beginning line 14, or type `b 14`)
8. In the window to the right, click "`Run`", and notice the program stops with a green arrow at line 14.
9. File->Open Source->tests.cpp
10. Set a breakpoint on line 12 (`b 12`). To delete a breakpoint, type "`delete X`" where `X` is the breakpoint number. To delete all breakpoints, type `delete`. For a list of breakpoint-related commands, type `help breakpoints`.
11. In the window to the right, click "`Cont`".
12. The program should stop at line 12 in tests.cpp.
13. Click `Next` twice.
14. Notice that we are about to run `testResult()`, on line 14.
15. `testResult` is a function, but which one? We could check in the previous file, or we could right click on `testResult()` in the source and "`display test result`" -- in the upper window, we see that it says `addEndEvent()`.
16. Click "`Step`" (*not* `Stepi`). Notice that the program jumped into the `addEndEvent()` function.
17. Notice that the green arrow now points to the first line in `addEvent()`.
18. Click Next until the arrow is on line 33.
19. Right click e1, and then "Display e1" Notice that it pops up in the top window.
20. You could double-click next to Description, and then next to `__r_`, then the top line of dots, but it is easier to type "`p e1`" and then locate the description ("Breakfast")
21. Right click on the "`d`" in `d.scheduleEvent(e1)` (line 33), then click `Display d`. Notice that you can see the pointer to the event list, the `eventCount`, and `eventCapacity`.
22. You can double-click on eventList to see it (as before). There aren't any events yet.
23. You could also type `p d.eventList[0]` to see the first event.
24. Click "`Next`" Notice that the program executed the `d.ScheduleEvent(e1)` and you can the result in the Event (breakfast was scheduled properly).
25. We want to step into `scheduleEvent(e2)` to see what is going on, so we open up `dDA-1.cpp` (student one's Dynamic Array file). Double-click on line 26, or (if that doesn't work, type "`b dDA-1.cpp:26`" (i.e., set a breakpoint at line 26 in file `dDA-1.cpp`)
26. click "`Cont`", and the program should break at line 26 in `dDA-1.cpp`.
27. click "`Next`" until you get to line 33 (`if (eventCount==0)`). Type `p eventCount`, and you should see `1`.
28. Click "`Next`" and the program will skip to line 39.
29. Keep clicking `Next` and notice that the green arrow skips over the `if()` loop because the event (lunch) is not before breakfast.
30. Notice that the function never schedules the event if it is after the other events.
31. This is the bug! Fix the bug in `dDA-1.cpp`
32. Re-make the program (from the terminal) and run it again. Notice that there is another bug!
33. Fix this bug (*note:* the program might not fail at this point. The error is that the student forgot the brackets [] on the delete in their expand section.
34. Go onto the next error: `Cancel first event failed`.
35. Find the `cancelEvent()` function in `dDA-1.cpp` (line 64). Set a breakpoint at that line by typing `b dDA-1.cpp:64`
36. Run the program, which should stop on line 64.
37. Click next until the return line, and notice that the return happens after the first event! This is a bug. Fix it and continue with further bugs.