COP4530 – Data Structures, Algorithms and Generic Programming
Recitation 2
Date: September 11, 2009

Lab objectives:

1) **Quiz**
2) **Set up SSH to run external programs.**
3) **Learn how to use the DDD debuger.**
4) **Learn to use the `list` STL.**

Setup tasks:

1. Under the Start menu, find the folder named X-Win32 and open the Xutils32.

2. Start the SSH client. When the shell opens, go to *Edit Settings* and then go to *Tunneling*.

3. Check the box that says *Tunnel x11 con* and click *OK*. Now it is possible to run text editors in separate windows from the SSH console. Fo emacs for example, just type "xterm emacs" followed by the file name to be edited, and a separate emacs window will appear.

4. Logon to your CS account.

5. Create a directory named **`cop4530`**.

6. Go into the **`cop4530`** directory and create a sub-directory named **`recitation`**.

7. Go into the **`recitation`** directory. Type the command **`pwd`**. You should see something similar to the following on the screen:

   `/home/majors/your_username/cop4530/recitation`

8. Type the following command,

   `cp –r ~cop4530/fall09/recitation/rect3 .`

   You should see some error messages similar to the following,

```
cp: cannot open
`/home/courses/cop4530/spring08/recitation/rect3/lists/readstudent.cpp' for
reading: Permission denied
cp: cannot open `/home/courses/cop4530/spring08/recitation/rect3/wcount-
good.cpp' for reading: Permission denied
```

   Note: The warnings above indicate that some files cannot be read because the permissions have been set to non-world viewable.

Task 1 : Learn to use the DDD debugger.

A debugger allows us to run other program executables and examine the behavior of these programs as they are running. Debuggers are especially helpful when there is a mysterious bug in the program that is not apparent at first glance.

One of the more popular debuggers for UNIX would be the GDB (the GNU debugger). A graphical front-end version of GDB is known as the DDD.

1. Open the file **`wcount-bad.cpp`**.

2. Compile the program and run the executable. The desired effect of the program should be as follows:

    i. The user is allowed to add enter 3 distinct words. If the word entered already exists, the count for the word is incremented for each repeated entry.

    ii. The program should not accept any new words if all 3 slots are full.

    The desired output for the program is as follows:

```
Please enter word (-1 to quit)
one
Please enter word (-1 to quit)
two
Please enter word (-1 to quit)
three
Please enter word (-1 to quit)
four
Array is full. Cannot add word.
Please enter word (-1 to quit)
one
Please enter word (-1 to quit)
three
Please enter word (-1 to quit)
-1
Exiting program..
wordarray[0] = one, count = 2
wordarray[1] = two, count = 1
wordarray[2] = three, count = 2
```

3. This program has logic errors. Use the DDD debugger to determine the location of the error.

4. Fix the error to produce the desired output above. *Hint: An unused function in the program will be helpful.*

A sample solution will be provided in the file **`wcount-good.cpp`**. This file will be made available by 3:00 p.m. today (January 23, 2008). You may obtain the file by typing the following in your current directory:

```
cp ~cop4530/fall09/recitation/rect3/wcount-good.cpp .
```

Exercise

Log on to `linprog.cs.fsu.edu` and use the DDD debugger to solve a bug problem with an executable created from the file **`squares-bad.cpp`** located in your **`rect3`** directory.

Task 2 : Learn to use the list STL.

1. Go into the lists directory.
2. Open the file readname.cpp. This client program uses the list STL to store in names keyed in by the user. When the user is done, the program prints out all the names stored in the linked-list.
3. Suppose you wish to write a similar program to print out some student information. This program should store the name, age and section of each student.
   i. Modify the program readname.cpp to prompt the user for the age and section corresponding to each name.
   ii. Store all three information about each student in a single linked-list.
   iii. Use a FOR or WHILE-loop to print out the information stored in the linked list.
4. A sample output for such a program is shown below:

```
Enter  name ('exit' when done): Susana
Enter  age: 19
Enter  section: 1
Enter  name ('exit' when done): Lucas
Enter  age: 21
Enter  section: 2
Enter  name ('exit' when done): Arturo
Enter  age: 23
Enter  section: 1
```



```
Enter name ('exit' when done): Jane
Enter age: 20
Enter section: 2
Enter name ('exit' when done): exit
Name = Susan, Age = 19, Section = 1
Name = John, Age = 21, Section = 2
Name = Bill, Age = 23, Section = 1
Name = Janet, Age = 20, Section = 2
```

Sample solutions will be provided in the file lists/readstudent.cpp. This file will be made available by 3:00 p.m. today (January 23, 2008). You may obtain the files by typing the following command in your current directory.

cp ~cop4530/fall09/recitation/rect3/lists/readstudent.cpp

Topic	Links
Templates	**http://www.cplusplus.com/doc/tutorial/tut5-1.html**
DDD Debugger	http://www.gnu.org/manual/ddd/html_mono/ddd.html