COP4530 – Data Structures, Algorithms and Generic Programming
Recitation 3
Date: September 12th, 2005

Lab objectives:

1) **Learn to use the `list` STL.**
2) **Learn how to use the DDD debugger.**
3) **Discussion and Q & A about Assignment 1.**
4) **Quiz.**

Setup tasks:

1. Logon to your CS account.
2. Create a directory named **`cop4530`**.
3. Go into the **`cop4530`** directory and create a sub-directory named **`recitation`**.
4. 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`

5. Type the following command,

   `cp –r ~cop4530/fall05/recitation/rect3 .`

   You should see some error messages similar to the following,

   ```
   cp: cannot open
   `/home/courses/cop4530/fall05/recitation/rect3/lists/readstudent.cpp' for
   reading: Permission denied
   cp: cannot open `/home/courses/cop4530/fall05/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 `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): Susan
Enter age: 19
Enter section: 1
Enter name ('exit' when done): John
Enter age: 21
Enter section: 2
Enter name ('exit' when done): Bill
Enter age: 23
Enter section: 1
Enter name ('exit' when done): Janet
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 2:00 p.m. today (September 12th, 2005). You may obtain the files by typing the following command in your current directory.

```
cp ~cop4530/fall05/recitation/rect3/lists/readstudent.cpp .
```

Task 2 : 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 12:00 p.m. today. (September 12th, 2005). You may obtain the file by typing the following in your current directory:

```
cp ~cop4530/fall05/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 : Discussion and Q& A about Assignment 1

Advice from TA:

1. **Read the project requirements carefully.**
   - You should attempt to satisfy all the requirements of the project.
   - Your executable will be tested with an automated script, so your program should behave in the same way as the executable.
   - Your program should also print out the output in the same manner as the executable.
   - When in doubt, clarify with instructor or TA. Do not assume.

2. **Start now if you have not done so.**
   You need time to test and debug your program.

3. **Plan ahead before you code.**
   Although this is a warm-up project, you should still plan ahead before your start. For instance, you should decide on how you need to store the data so that it can be efficiently accessed when needed.

4. **Familiarize yourself with the reading and parsing input.**
   This is one of the more fun parts of the project. You get to practice/hone some of your programming skills in parsing input.

5. **Test your program extensively and write your own test files.**
   Do not presume that your program will only be tested with the sample input data already provided.

6. **Use I/O redirect to test your program with an input test file.**
   A sample file **`input.txt`** is provided to you. You can test your program by typing,

   ```
   proj1 FLIGHTS.txt < input.txt
   ```

7. **Seek help from instructor or TAs if you are stuck for too long.**
   If you have been staring at the same segment of code for the past 3 hours and still can't figure out what is wrong, you need help! Submit your current work using the submission script and email me at toh@cs.fsu.edu with a brief description of your problem.

8. **Adhere to good programming styles and practices** as your program will be graded on design and readability (see Syllabus),

   e.g. indent your code, include comments on major subroutines or algorithms (but not on every line of code), close all open file descriptors, and etc.

9. **Do your own work.**

Q & A: ?

Task 4 : Quiz

Please answer the questions in the quiz handout.

References

Topic	Links
Templates	1. http://www.cplusplus.com/doc/tutorial/tut5-1.html
DDD Debugger	1. http://www-users.itlabs.umn.edu/classes/Fall-2002/csci1113/ddd_handout.pdf 2. http://www.gnu.org/manual/ddd/html_mono/ddd.html