CPSC 2430 Data Structures

Winter Quarter 2024

Lab 1 **Due: 10:00pm, Jan 09, 2024**

Lab #1 is a warmup lab. In this lab, we will first set up the lab/assignment environment on the department Linux server. Then, we will implement a C++ function with a focus on IO, controls, variables, and functional decomposition.

Task 1: Set up your environment on the server.

Step 1. Check your account

In this course, all the labs and projects will be done on the department Linux server **cs1.seattleu.edu**. If your account has been established, you can login to the server using your SU account name and password with a telnet program (**SSH Client** or **Putty**).

Start **SSH client** (or **Putty**), and login to **cs1.seattleu.edu** using your SU account name and password.

In order to do the lab and assignment projects from your home or dorm, you need to install a SSH client (or Putty) on your laptop/desktop. Then you can access the server **cs1.seattleu.edu** as described above. Windows 10 and Mac OS have a built-in SSH client that you can use: Windows Terminal and Terminal, respectively. Open a terminal and type:

```
ssh username@cs1.seattleu.edu
```

If you want to use Putty, visit http://www.chiark.greenend.org.uk/~sgtatham/putty/ to learn and download Putty. A tutorial for Putty for Mac can be found https://www.ssh.com/academy/ssh/putty/mac and Putty for Windows 10 can be found https://www.ssh.com/academy/ssh/putty/windows.

Step 2. Set up your directory

After you log in to the cs1 server, you are now in your home directory. Create a directory for CPSC 2430 named **cpsc2430** by typing the following command at the prompt:

```
mkdir cpsc2430
```

If the directory is correctly made, there will be no response from the system except the return of your prompt.

Now see if the directory cpsc2430 is successfully created, type the following command at the prompt:

```
ls
```

If the directory is correctly created, you will see it on the list.

Now visit the newly created directory by typing the following command at the prompt:

```
cd cpsc2430
```

Now are you in the cpsc2430 file directory. Using the same approach, you need to create a directory for Lab 1 and visit the Lab 1 directory:

```
mkdir lab1

cd lab1
```

Anytime when you want to go to an upper-level directory, type the following command at the prompt:

```
cd..
```

Step2 example:

```
Last login: Fri Sep 17 10:10:10 2024 from 50.46.233.13
[mjilani@cs1 ~]$ pwd
/home/fac/mjilani
[[mjilani @cs1 ~]$ mkdir cpsc2430
[mjilani@cs1 ~]$ ls
cpsc2430
[mjilani@cs1 ~]$ cd cpsc2430
[mjilani@cs1 cpsc2430]$ mkdir lab1
[mjilani@cs1 cpsc2430]$ ls
lab1
[mjilani@cs1 cpsc2430]$ cd lab1
[mjilani@cs1 lab1]$ pwd
/home/fac/mjilani/cpsc2430/lab1
```

```
[mjilani@cs1 lab1]$
```

Step 3. Program creation, compilation, and execution on the server.

Step 3.1. Create and edit a program using emacs.

To create a new C++ program (suppose we want to create a file named test.cpp), type

```
emacs lab1.cpp
```

You should see the textual edit interface of emacs.

Now type the following code to test.cpp:

```
#include <iostream>
using namespace std;
int main(){
        cout << "This is a testing program." << endl;
}
```

Don't forget to **save your program!!!** To save your program, press ***ctrl+x*** first, then press ***ctrl+c***. A prompt message will show up at the bottom to ask you if you want to save the file, type ***y*** to save the file.

If you want to edit your test.cpp, type the command again:

```
emacs lab1.cpp
```

If you simply want to review your program, type:

```
cat lab1.cpp
```

Step 3.2 Compile your program.

To compile your program, type the following commands:

```
g++ -Wall -Werror -pedantic -std=c++11 -o lab1 lab1.cpp
```

"-Wall" is short for "warn all" and will turn on most compiler warnings

"-Werror" is short for make all warnings into errors

"-pedantic" issues all warnings demanded by strict ISO C++ rules for extra safety

"-std=c++11" specifies the C++ version, in this case C++ 11

"-o <target_name>" to specify the name of the target (e.g. "lab1", which will be the executable program). If you don't specify a target, your executable will be named **a.out**

If errors and/or warnings occur during the compilation process (they will be listed on the screen), you will need to edit your program and make corrections. "No news is good news" – in other words, if you simply see the Linux prompt again after compiling, it means your program is compiled successfully.

Step 3.3 Run your program

To run the executable main, type the following command at your Linux prompt:

```
./lab1
```

You should see the output of this lab1.cpp ("This is a testing program."). For example:

```
[mjilani@cs1 lab1]$ cat lab1.cpp
#include <iostream>
using namespace std;
int main(){
  cout << "This is a testing program." << endl;
}
[mjilani@cs1 lab1]$ g++ -Wall -Werror -pedantic -std=c++11 lab1.cpp -o lab1
[mjilani@cs1 lab1]$ ./lab1
This is a testing program.
[mjilani@cs1 lab1]$
```

Step 3.4 Submit your code

For all coding labs and assignments you will be required to submit your code using the script provided by the instructor for each. Try submitting your lab1.cpp by running:

```
/home/fac/mjilani/submit/24wq2430/lab1_submit
```

Task 2: Write a function *bico(int n, int i)* that returns the i^{th} coefficient of binomial $(x + y)^n$. For example, bico(4,2) should return 6 because

$$(x+y)^4 = x^4 + 4x^3y + 6x^2y^2 + 4xy^3 + y^4.$$

The second coefficient of $(x + y)^4$ is 6. Notice that coefficient index starts from 0. Your program needs to receive two integers from user input. The first one refers to the degree of this binomial and the second one refers to the coefficient we want to obtain.

For this lab, you must use Pascal's triangle to find the desired binomial coefficient. **You must calculate the coefficients from previous Pascal triangle values. You are not allowed to use any formulas, combinatorial or other, to calculate the coefficients.**

Your program must use dynamic memory allocation to store the Pascal Triangle up to the degree requested by the user. Your program should have an interface like the one below to receive input from users. After input is received, **your program must print the Pascal Triangle up to the degree specified like below, AND print out the requested binomial coefficient value.**

```
[mjilani@cs1 lab1]$ g++ -Wall -Werror -pedantic -o lab1 lab1.cpp
[mjilani@cs1 lab1]$ ./lab1
Please input the degree of the binomial: 4
Please input the index of the coefficient: 2
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
The result is: 6
```

Input

Make sure to TEST your program with different binomial coefficient degree/index combinations and check your result is correct.

Lab 1 Submission

You need to submit the following file :

- lab1.cpp

Before submission, you should ensure your program has been compiled and tested (extensively). Your assignment receives zero if your code cannot be compiled and executed. Your cpp file should compile in cs1 server with g++ and all prescribed flags with no errors and produce an executable file.

You can submit your program multiple times before the deadline. The last submission will be used for grading.

To submit your assignment, run the script below from the directory containing your lab1.cpp file (assuming your files are on cs1.seattleu.edu):

```
/home/fac/mjilani/submit/24wq2430/lab1_submit
```

Lab 1 Grading Breakdown

Breakdown	Points	Note
Set up your environment	20	Set up your cpsc2430 environment on the server.
Lab 1 submission	10	All the required files are submitted.
Compilation	10	Your file compiles with no errors and generates an executable file.
Functionality	50	Receive input from user: 10 pts Print Pascal Triangle correctly: 20 pts Return the coefficient correctly: 20 pts
Coding	10	Clean, well-commented code. No unsolicited output messages, such as testing & debugging messages. All dynamically allocated memory is freed up accordingly. No use of global variables and proper functional decomposition.

Some resources you may need:

1. Basic SSH commands: https://www.hostinger.com/tutorials/ssh/basic-ssh-commands
2. Emacs tutorial: https://www2.lib.uchicago.edu/keith/tcl-course/emacs-tutorial.html.