SCSSE, University of Wollongong
CSCI204/MCS9204 Object and Generic Programming in C++
Laboratory Exercise Week 2

Task 1: Getting started

1. Login into Banshee:
    a. Boot the PC into Ubuntu Mode:
        i. Click "Console" to open a terminal;
        ii. Type the command
        ssh *your-login-name*@banshee.cs.uow.edu.au
        and input your password.
    b. Boot the PC into Windows mode:
        i. Click the icon "SSH Secure Shell Client" on the Desktop, or Click "Start" → "All Programs" → "SSH Secure Shell" → "Secure Shell Client".
        ii. Click the button "Quick connect" on the prompt window of SSH Secure Shell, enter "banshee" (or "banshee.cs.uow.edu.au" when you run SSH Secure Shell at home) in the text field for "Host Name", enter "your-login-name" in the field for "User Name", then click the button "Connect". It is your choice to save a new host key to the local database or not. You may choose "No" when you use the lab computers, "yes" when you use your own laptop. Then enter your password and click "OK".

2. You should now be logged into your home directory. Type the following:
   `ls -la .profile`
   If the response is: `\.profile: No such file or directory` you should type:

   `cp /share/cs-pub/204/.profile .`

   This profile sets the shell and some other useful functionality.
3. If you are unfamiliar with Unix you can get a brief summary of some useful commands in *Linux User Guide* and *Lab Tips* located in the section Resources.
4. If you have already made a directory 204, either in your home directory or in a subdirectory if you have an area for keeping subject work. You don't have to do this but it keeps things tidy.
5. Change into the directory 204 and make a new directory in it called Labs.
6. Change into the directory Labs and make a new directory in it called Lab1.
7. You should place your solutions for this Lab in this directory.
8. You can open files using a text editor (foe example *nedit* ...).
9. Future lab work should be placed in similar directories.

When you work at home you will probably use SSH file transfer. See SSH related guides on the subject web site in the section Resources.

Task 2: Arrays and pointers

Write two C++ programs which implement the following:

1. Declare an array of three integers.
2. Implement a loop that reads three values from the standard input stream into the array.
3. Implement another loop that outputs the three values stored in the array into the standard output stream.

In the first version, **t1a.cpp**, you need to use array indexes. In the second version, **t1b.cpp**, you need to use pointers.

Task 3: Functions and reference variables

Write two C++ programs to determine the cost of building a square table. In the first version, **t2a.cpp**, you should use functions with parameters passed by value. In the second version, **t2b.cpp**, you should pass all parameters by reference with no return types (void). The programs should use seven functions:

1. A function that gets from the standard input stream the number of chairs for the table.
2. A function that gets from the standard input stream the surface area (in m^2) of the table.
3. A function that gets from the standard input stream a colour for the cushions on the chair seats. Any single word colour should be accepted.
4. A function that accepts from the standard input stream the type of wood used to build the table and chairs; 'm' for mahogony, 'o' for oak or 'p' for pine. Any other type entry should be rejected.
5. A function that takes the number of chairs, the surface area of the table and the type of wood and calculates the price. The price for a table with surface area S and N chairs is x(S + ½ N) where x is $200, $150 and $95 respectively for mahogony, oak and pine.
6. A function to display the details of the purchase.
7. A function to display the price of the purchase.

An example of the expected input/output is shown in **Sample_T3.txt**.

Task 4: Basic class/object construction

Create a program **account.cpp** that defines a class named `Account`. It shall contain data fields to store the account number, the account type and the balance. The class shall also contain the following member functions:

- `setAccount( )` that takes three parameters – the account type, the account number and the balance, and stores these values in the corresponding data fields
- `getAccType( )` that returns the account type
- `getBalance( )` that returns the balance
- `getNumber( )` that returns the account number
- `withdraw( )` that takes a parameter - the amount to be taken from the account and modifies the corresponding data field. The function should return `false` if the specified amount of money is not available on the account and true otherwise.

Consider a simple scenario and provide an example of the main( ) function that implements this scenario.