**LAB 1**

**Purpose**

This lab should help you get started with C++.

**Key Reading**

* Appendix D, page 493

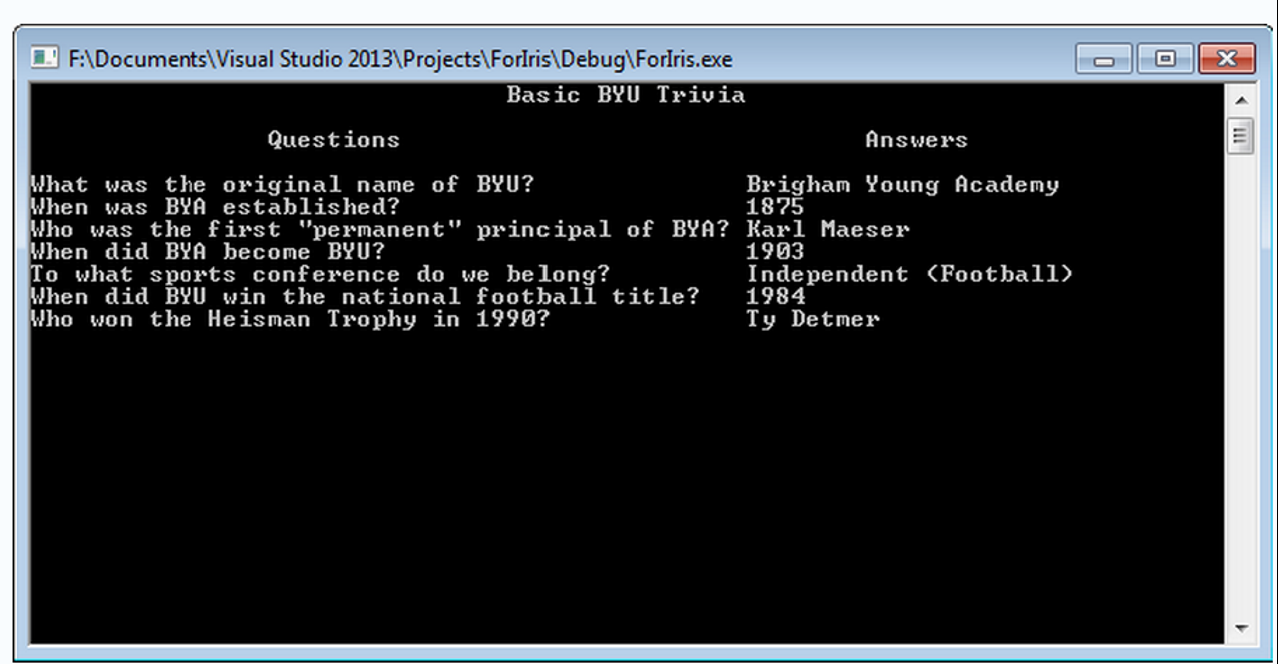
**Requirements**

**Basic BYU Trivia (8 points)**

* Recreate the following output **exactly** (use escape sequence tabs to align the columns).**No endl allowed.**
* Use escape sequences for new lines, tabs, and quotes at least.
* NOTE: The 5th question's answer has **parentheses** around Football.
* You ARE allowed to have system("pause") in your program.

**Good Coding Style (2 points)**

* Does your code exemplify the good coding habits taught in class?



## LAB 2

## Purpose

This lab introduces some basic concepts and techniques of the C++ programming language. Master these concepts and techniques because they will be used time and time again in future lab assignments. This lab also introduces the C++ reference website with its predefined functions.

## Key Reading

* 1.7
* 2.1-2.4

## Background

After an exhausting campaign, you've been elected as President of the Pizza Council of BYUSA. Your primary responsibility is determining how many pizzas should be ordered for each BYUSA event. To make things easier, you decided to write a C++ program to help you do the calculations.

## Requirements

### Part 1 - Count Your Many Pizzas (4 points)

* Prompt the user for the number of guests attending the event.
* Determine and report the number of large, medium, and small pizzas you need to order.
  + For every 7 guests, order one large pizza
  + For every 3 guests left over, order one medium pizza
  + For every 1 guest left over, order one small pizza

### Part 2 - Serving Size (4 points)

* Compute and report the total area of pizza (in square inches) you need to purchase. Do**not** round these values. See table below for examples.
* Compute and report the total area of pizza (in square inches) each guest can eat. Do **not** round these values.

### Part 3 - Supplementing the Budget (4 points)

* Prompt the user for the percent of the total price to be paid as a tip.
  + You can assume the tip percentage will be input as an integer from 0 to 100
* Compute and report the total cost (including tip) of all the pizzas, **rounding** to the nearest dollar. ***Note***: Changing the value type into an int alone will not round to the nearest dollar.

### Good Coding Style (3 points)

* Does your code exemplify the good coding habits taught in class?

## Requirement Notes

### Requirements Before Pass Off

* Use const double PI = 3.14159 as π (pi) as C++ does not have a predefined value for π. The examples below were computed using these values.
* You **must**use constants for what otherwise might look like magic numbers (radius, pi, costs, 7, 3, 1, etc).
* Create **your own** 3 test cases by computing the output manually with a calculator and then comparing it with the output of your program. Make sure to include these test cases in a comment block at the top of your code.

### Requirements For Program

The following information is drawn from the menu of Jenga's Pizza:

|  |  |  |
| --- | --- | --- |
| **Pizza** | **Price** | **Diameter** |
| Large | $14.68 | 20 inches |
| Medium | $11.48 | 16 inches |
| Small | $7.28 | 12 inches |

The following table shows some example input and output (input is bolded):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **People** | **Tip** | **Larges** | **Mediums** | **Smalls** | **Area (in2)** | **Area/Person (in2)** | **Cost ($)** |
| **2** | **15%** | 0 | 0 | 2 | 226.194 | 113.097 | $17 |
| **4** | **10%** | 0 | 1 | 1 | 314.159 | 78.5397 | $21 |
| **12** | **8%** | 1 | 1 | 2 | 741.415 | 61.7846 | $44 |
| **48** | **17%** | 6 | 2 | 0 | 2287.08 | 47.6474 | $130 |
| **120** | **25%** | 17 | 0 | 1 | 5453.8 | 45.4483 | $321 |
| **305** | **20%** | 43 | 1 | 1 | 13823 | 45.3213 | $780 |

# **LAB 3**

# Purpose

This assignment introduces you to conditional statements and boolean expressions.

## Key Reading

* 2.3
* 3.1-3.2

## Background

Hybrid cars usually have higher initial costs but lower fuel costs than non-hybrid cars. To help a car buyer decide which car has overall lower costs, you can to write a program to compute the cost of owning a car for 5 years. A user could then run your program for two potential car purchases in order to compare the resulting costs.

## Requirements

### Part 1 – User Input (8 points)

Prompt the user for the following input:

* The estimated miles driven per year
* The estimated price of a gallon of gas during the 5 years of ownership
* The initial cost of a hybrid car
* The efficiency of the hybrid car in miles per gallon
* The estimated resale value (a dollar amount) for a hybrid after 5 years
* The initial cost of a non-hybrid car
* The efficiency of the non-hybrid car in miles per gallon
* The estimated resale value (a dollar amount) for a non-hybrid after 5 years
* The user's buying criterion, either  minimized "Gas" consumption or minimized "Total" cost. Assume the user will enter one of these two strings

Make sure that you prompt in **this same** order. If you do not, you will be marked off.

Verify that all numerical input are positive. You will need to check each number individually and reprompt **immediately**in the event of bad numeric input. Output a message that only positive numbers are valid and reprompt one more time. You may assume that the second value will be a positive number.

### Part 2 – Output Costs (8 points)

For each car, your program should output the following:

* A label indicating whether the car is Hybrid or Non-Hybrid
* The total gallons of fuel consumed over 5 years
* The total cost of owning the car for 5 years (fuel cost + depreciation in car value)

For the last question you prompt for, if the user's criterion is "Gas", then provide the output for the car with the lower gas consumption first.  If the criterion is "Total", then provide output for the car with the lowest total cost first. For every case, you will need to provide output for both vehicles, each with the total cost and the total gallons used.

### Good Coding Style (4 points)

* Does your code exemplify the good coding habits taught in class?

## Requirement Notes:

* No magic numbers (any numeric literal, such as 5 or 3.99 is a magic number)
* 3 test cases written in a comment block at the top of your code
* You must prompt the user in**the same order** as listed in Part 1

## Extra Credit (1 point)

Assume that

* A new, non-hybrid car costs $15,000 and gets 25 mpg
* A new hybrid car costs $32,000
* You resell any car after 5 years for 75% its original value
* You drive 10,000 miles in a year
* Gas costs $2.50

What gas mileage would a new hybrid need to have in order for the total costs over 5 years to be equal to the total costs for a non-hybrid car? Experiment with different values using your program to find the correct answer.  Then write your answer in a comment at the top of your code.

**LAB 4**

## Purpose

This lab introduces you to loops. These control structures facilitate powerful programs. It is important that you become comfortable with the various kinds of loops, as each has its own uses.

Nearly all of this lab can be done without loops of any kind. However, you should use loops as often as possible in this assignment. They will make your code easier to read and debug and will prepare you for future assignments.

## Key Reading

* 3.7
* 4.1-4.3, 4.9

## Background

Your best friend bought tickets for the both of you to go to The Price Is Right this weekend. On the extremely small chance that your name gets called, you want to be ready to win as much money as possible. Instead of preparing for a wide variety of games, though, you put all of your effort into the one game that truly matters: Plinko.

To practice your Plinko skills, you've decided to write a C++ program to simulate a game of Plinko and to compute average winnings based on where you drop your chips. We assume the plinko board looks exactly like the one shown below.

## Requirements

### Part 1 - The Menu (8 points)

* Prompt the user to select one of three operations **(your program must use the options 1, 2, and 3)**:
  + **1 -**Drop a single chip into one slot
  + **2 -**Drop multiple chips into one slot
  + **3 -** Quit the program
* After any operation is complete (except for quitting), reprompt the user to select another operation (return user to your menu)
* If an incorrect option is entered, you must reprompt the user to select another operation (return user to your menu)
* For Part 1, only the quit operation needs to work

### Part 2 - Let the Chips Fall (12 points)

* Allow the user to drop one chip into one slot
* Prompt the user to select a slot into which he or she will drop a chip
  + If the specified slot is not between 0 and 8, **immediately**return the user to the menu
* During simulation, report the location of the chip as it falls
* Report the amount of money won for the chip

### Part 3 - Bowl of Chips (12 points)

* Allow the user to drop any number of chips into one slot
* Prompt the user to enter the number of chips to drop
  + If the number of chips is nonpositive (including 0), **immediately**return the user to the menu
* Prompt the user to select one slot into which he or she will drop the chips
  + If the specified slot is not between 0 and 8, **immediately** return the user to the menu
* Report the total and average amount of money won for all chips
* Do **NOT** report the location of each chip as it falls down the plinko board.

### Good Coding Style (8 points)

* Does your code exemplify the good coding habits taught in class?

## Requirement Notes

### General

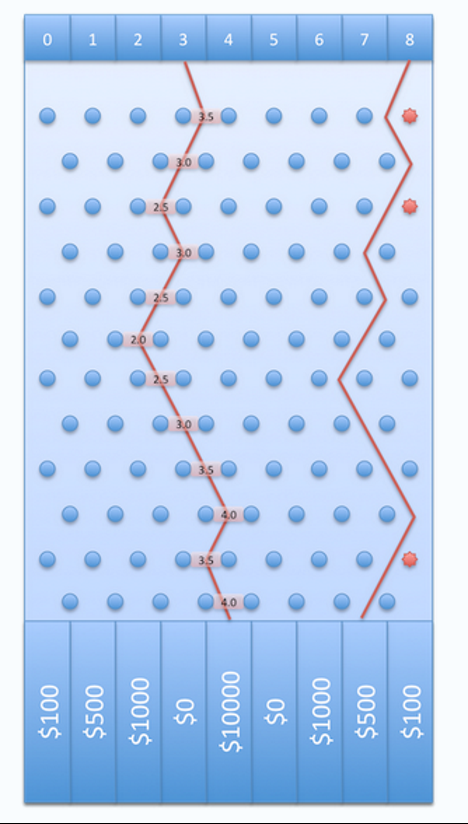
* If you are not familiar with Plinko, you may want to play it a bit before beginning the lab. An online version of Plinko can be found [here](http://www.kongregate.com/games/StapleGun/plinko).  Note that the dimensions, prizes, or other details may be different in the demo from the requirements of this lab. **Use the values specified below, not those found in the demo.**

### Requirements for Pass Off

* Must include the random seed, srand(time(0)), in the main function.
* No Magic Numbers (prize money, etc) , except for the slots.
* 3 test cases in a comment block at the top of your code.
* Chips must not fall off the board (chip location must never be less than 0 and must never be greater an 8).
* ANY invalid input under ANY option should return the user to the menu, not reprompt for a good input.

### Plinko Board

* A Plinko board consists of 9 input slots and 9 output slots. The money won for a chip is determined by the slot the chip falls into.
* Between the input and output slots, there are 12 rows of pegs, with each row's pegs centered directly under the spaces from the row of pegs above.
* Upon encountering a peg, a chip has a 50/50 chance of going left or right (although chips must always stay on the board).
* The accompanying image demonstrates the board, its slot numbers, its prizes, and two sample paths.
* The left path displays the position of the chip at each step; the right path shows red pegs whenever the left (or right) decision is forced.
* For the left path in the example, the following is the expected location report: [3.0 3.5 3.0 2.5 3.0 2.5 2.0 2.5 3.0 3.5 4.0 3.5 4.0]



**LAB 5**

## Purpose

This lab introduces you to functions, the backbone of many programming languages. Functions greatly simplify and organize code. **Your solution to this lab must use functions appropriately.**

## Key Reading

* 5.1-5.4

## Requirements

### Part 1 - Plinko with Functions (8 points)

* Refactor your Lab 4 code (i.e. preferably do not rewrite Lab 4 from scratch) using functions. See the requirement notes (below) for an explanation of the difference between refactoring and rewriting. You may take either approach to completing this lab.
* In addition to using functions effectively, your Lab 5 solution must behave the same as your Lab 4 solution. In other words, for Part 1 of this lab you must follow all requirements from Lab 4 and complete all parts of Lab 4, but this time using functions.

The following are descriptions of the required functions the TAs will expect to see during your Pass-Off. We ask you to write these functions in hope that you will call them more than once.

* Simulating one chip falling
* Simulating multiple chips falling
* Getting Prize Money

### Part 2 - **Functions and the Power of Code Reuse**(8 points)

* In this part of the lab, you will take advantage of functions to add an additional menu option to your Plinko program, **without having to duplicate code from the other menu options.**
* Add a menu option that allows the user to drop the same number of chips into each of the 9 slots (i.e., if the user enters 5, you simulate dropping 5 chips into each of the 9 slots).
  + Prompt the user to enter one number, which is the number of chips to drop into every slot.
  + If the number of chips is non-positive, **immediately**return the user to the menu.
  + For **each** slot, report the total and average amount of money won for all chips dropped into that slot.
  + Do **NOT** report the location of each chip as it falls through the plinko board.

### Good Coding Style (4 points)

* Does your code exemplify the good coding habits taught in class?

## Requirement Notes

### General

* The TAs will read your code on this lab to see how you are using functions. Merely adding a few arbitrary functions is insufficient to complete this lab. **You also should have minimal code duplication.**
* You must **not** use Global Variables.

### **Requirements For Pass Off**

* You should **NOT** use global variables on this lab. You'll need to pass parameters to functions and return results.
* 3 test cases must be included in a comment block at the top of your code.
* You should test your code for all Lab 4 Requirements again.
* No Magic Numbers (prize money, etc) except for slot numbers.

### **Refactoring vs. Rewriting**

* To complete Part 1 of this lab, you may refactor your Lab 4 code. Refactoring means reworking source code without changing its functionality. Programmers refactor when they see ways to improve the maintainability (i.e., readability, extendibility, etc.) of source code.
* Before refactoring, you are **strongly** encouraged to copy your Lab 4 code into a new solution for Lab 5 (rather than to edit your Lab 4 code directly). This way, if your Lab 5 code gets too scrambled, you still have your Lab 4 code from which to start over.
* If you get stuck trying to refactor, or refactoring seems to complicated, you can rewrite the Plinko program from scratch using functions.

**LAB 6**

## Purpose

Gain experience with 2-dimensional arrays and loops while solving a real-world problem ("steady state analysis").

## Key Reading

* **4.3**
* **4.8**
* **5.1-5.9**
* **6.1-6.6**
* **8.1**
* **8.3**

## Background

In this assignment you will compute the steady state temperature distribution over a piece of metal. This steady state analysis is used in solving many physics problems. Similar matrix algorithms are also used in analyzing airplane wing and automobile pressure. These types of calculations ensure that everything works correctly. Though the full analysis may be complicated, pieces of it are quite simple, such as the problem described below.

## Requirements

### Part 1 - Initialize and Print 2D Array (4 points)

The problem is to determine the temperature distribution for a two-dimensional plate with constant boundary conditions. Use a 20 X 20 two-dimensional array of numbers to represent the temperature at different places on the plate. The elements on the boundaries have fixed temperatures. The elements on the top and bottom row have a fixed temperature of 100 degrees. The elements on the leftmost and rightmost columns have a temperature of zero degrees. (The corner elements have a temperature of zero degrees.)

**You must print the initialized 2D array to the screen.**

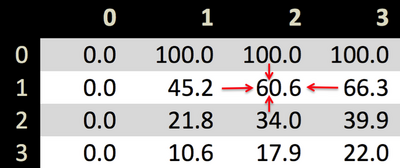
An initialized 4 X 4 matrix would look like this. You must initialize all squares not on the edge to 0:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 100 | 100 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 100 | 100 | 0 |

### Part 2 - Update Elements Once (4 points)

The job of the algorithm is to find the steady-state temperature distribution of the interior elements which are constantly changing to become the average of the temperature of their neighbors. Steady state means that the temperature of any cell is virtually unchanging and approximately equal to the average of the temperatures of its four neighbors on the north, south, east, and west directions. Note that the steady state temperatures will be the same regardless of what the beginning temperatures are for the interior elements.

To calculate the new temperature value of an array element, take the average of the values in the 4 neighbors of that cell from the previous iteration.  This value will be placed in the output array.

[](https://facwiki.cs.byu.edu/cs142fa11/index.php/File:Heatplate.png)

For example, the cell at position array[1][2] currently has a value of 60.6 degrees. At the current iteration, the element in the output array will be set to the average of its 4 CURRENT neighbors (100.0 + 66.3 + 34.0 + 45.2)/4.

Print the first iteration.

**NOTE: MAKE SURE TO WORK ON THE TWO 2D ARRAYS, ONE OF WHICH HAS THE TEMPERATURES FROM THE LAST ITERATION AND THE OTHER CONTAINING THE NEW, AVERAGED TEMPERATURES.**

**FOR THE NEXT ITERATION, THE NEW, AVERAGED TEMPERATURES BECOME THE INPUT TEMPERATURES.**

### Part 3 - Repeat update until Stable (8 points)

You should continue to iterate until no cell in the array changes more than 0.1 degree, calculating the temperature for all interior cells on each iteration. Your program should monitor the largest change for any cell in the array in order to determine when to stop iterating (See Common Loop Algorithms in section 4.7 of the book for ideas).

### Part 4 - Using Excel to Display Results (4 points)

When you are finished, you should write your two dimensional array out to a comma separated values file like this:

0.0,100.0,100.0,100.0,

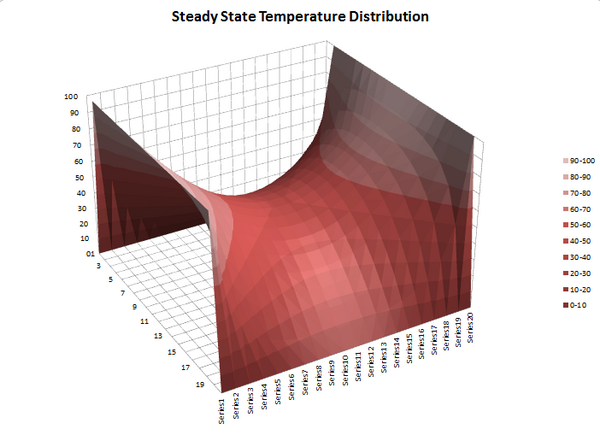
0.0,45.1,60.6,66.2,0,

0.0,45.1,60.6,66.2,0,

0.0,100.0,100.0,100.0,0,

Essentially, all you need to do is write to a file that's named with a .csv extention. Look to the book for help in writing to a file and using ostream objects.

You can import this csv file into a spreadsheet like excel and format it as a 3D graph in order to check your work. If you've done your calculations correctly, your graph will look like this.

[](https://facwiki.cs.byu.edu/cs142fa11/index.php/File:Hotplate.png)

### Good Coding Style (5 points)

* Does your code exemplify the good coding habits taught in class?

## Requirement Notes

**Make sure you use functions and two 2D arrays (not vectors).**

To graph your data, import the csv file into excel as a csv file, then select the data and create a 3D chart. To change the rotation of the chart, right click on the image and the click on 3D rotation. Change the values of X and Y. Follow this tutorial if you're unfamiliar with Excel: [Export To Excel](https://facwiki.cs.byu.edu/cs142fa11/index.php/Export_To_Excel)

Some students find that Xcode doesn't create a file when they use ofstream. To fix this, do the following

* First, in the menu along the top of the screen, go to "Project>>Edit Scheme" and change the build type from debug to release.
* Second, hard-code the path, don't just use a filename. For example, write "/Users/[user name]/Desktop/hotplate.csv" instead of "hotplate.csv".
* If you're still having problems, make sure the stream is closed when you're done with it (e.g. myfile.close)
* You must **not** use Global Variables.

## Requirement For Pass Off

* Students must implement at least the following functions
  + Initialize Hot Plate
  + Average Hot Plate (one time through the 2D array)
  + Print Hot Plate
  + Export Hot Plate
* No Test Cases are Required for this Lab.
* No Magic Numbers (rows, columns, .1 , etc)