SYDE 121
Lab Number 5

Exercise 1: Diamond of Asterisks

Learning Objectives: Practice developing algorithms for more complicated loops.

Read This First

The objective of this exercise is to print out a diamond of asterisks. For example, given an input value of seven rows, the following would be displayed:

```
   *
  ***
 *****
*******
 *****
  ***
   *
```

What to Do

Ensure that the user enters an odd positive number less than 20 to represent the number of rows (do not simply exit if the user does not enter a proper value). Use only output statements that print either a single asterisk (*) or a single blank space, and use only 'for' structures for looping.

A conventional (brute-force) approach would be to write two separate loops, one to print the top half of the diamond, and another to print the bottom half of the diamond. However, a more elegant approach would use a *single* loop structure to print the entire diamond. For full marks, design an algorithm that uses a *single* loop structure (that would count through all the rows), rather than two separate loops. Note that the single loop structure can contain nested loops.

Hint: You may find the abs() function useful. This function takes an integer argument and returns an integer i.e.

```
int c = abs( a );
```

where integer c is assigned the absolute value of integer a.

What to Hand In

Email your code to the course account and submit a hard copy to the submission box.

Exercise 2: Payroll

Learning Objectives: Learn how to use the switch statement.

Read This First

A company pays its employees as managers (who receive a fixed weekly salary), hourly workers (who receive a fixed hourly wage for up to the first 40 hours they work and "time-and-a-half" ie. 1.5 times their hourly wage, for overtime hours worked), commission workers (who receive $250 plus 5.7% of their gross weekly sales), or pieceworkers (who receive a fixed amount of money per item for each of the items they produce – each pieceworker in this company works on only one type of item).

What to Do

Write a program to compute the weekly pay for each employee. You do not know the number of employees in advance. Each type of employee has its own code – ensure this information is conveyed to the user. Use a **switch** to compute each employee's pay based on the user entered employee's paycode. (Note: enum types are optional but keep in mind that **cin**, just like **cout**, is not programmed to directly handle enum datatypes.) Within the **switch**, prompt the user to enter the appropriate facts your program needs to calculate each employee's pay based on that employee's paycode. Ensure that the default case for the switch statement is an error condition.

After all the employees have been entered, display a table indicating the total number of employees and the total salary at each position as well as the overall totals across all positions.

What to Hand In

Email your code to the course account and submit a hard copy to the submission box.

Exercise 3: Determining pi

Learning Objectives: Set up a loop using a tolerance.

Read This First

The value of pi can be determined using the following infinite series:

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \ldots$$

Infinite series are a means of determining mathematical terms such as pi, sine, cosine, etc. algorithmically, for example, within a calculator. You will learn more about these types of mathematical expressions in your calculus courses.

What to Do

Display a table that shows the value of pi approximated by the first term of this series, followed by the pi approximated by the first two terms, etc. Indicate both the term number and the approximation to pi on the same line. Set your approximation of pi to 12 decimal places for display purposes.

Since this is an infinite series, you will need a stopping criterion. Each additional term in this infinite series becomes smaller in magnitude relative to its previous term. Set a tolerance value that represents the difference of the pi at the n^{th} step and pi at the $(n-1)^{th}$ step. You will need to use the math library for the function fabs( ) (the absolute value of a double expression). This function takes a single double expression as an argument and returns a double value i.e.

```
double c = fabs( a );
```

will allow c to take on the absolute value of a. Allow the user to set the tolerance and constrain this value to:

$$0.0 < \varepsilon < 0.1.$$

What to Hand In

Email your code to the course account and submit a hard copy to the submission box.

Due Date

This lab is due Friday, October 14 by 6:00pm.