

# CS2211a Assignment 4

Issued on: Thursday, November 12, 2015

**Due by: Thursday, 11:55 pm, November 19, 2015**

- For this assignment, only electronic submission at owl.uwo.ca is required.
- ONLY user **Courier New** (*size = 11 pts.*)
- ***Start each question in a NEW PAGE***
- ***Write the question number in a separate line followed by an empty line***
- After finishing the assignment, you have to do the following:
  - ❖ Type your report and convert it to the **PDF** format (*no handwriting*),
  - ❖ The report should include:
    - Answers to *all* questions/requirements in the assignment
    - Enough test cases (as well as sample outputs) to demonstrate and cover all possible options in your program
    - A copy of *all programs* that you have written
    - *Three flowcharts*, one for each question
  - ❖ Prepare a soft-copy submission, including:
    - A copy of your *typed* report
    - All programs that you wrote (*each program in a separate text file*)—*3 in total* (use meaningful program names). These program files **MUST BE** text ASCII files. Do not submit them as PDF or Word.
    - **A readme file to say what is the full command to be used to compile each program.**
  - ❖ Upload the soft-copy submission file-by-file (*5 in total*), or as an archived directory.

**Failure to follow the above format may cost you 10% of the total assignment mark.**

- Late assignments are strongly discouraged
  - 10% will be deducted from a late assignment (up to 24 hours after the due date/time)
  - After 24 hours from the due date/time, late assignments will receive a zero grade.

## Program 1 (35 marks)

The area of a circle of radius  $r$  is given by

$$\text{Area of a circle} = \pi \times r^2.$$

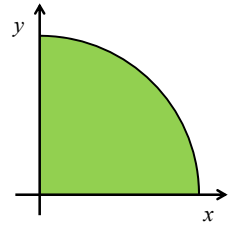
Imagine that you divided this circle exactly into 4 quadrants. The area of one quadrant is then  $0.25 \times \pi \times r^2$ .

Let us set the radius of this circle to be  $r = 1$ . The equation hence becomes

$$\text{Area of one quadrant of a circle of radius } 1 = 0.25 \times \pi.$$

We can simply say:

$$\pi = 4 \times \text{the area of one quadrant of a circle of radius } 1.$$



To calculate the area of the shaded quadrant of the circle, use a random number generator and guess effectively the correct value of the constant  $\pi$ . The technique you should use is to:

- Generate a random number between 0.00 and 1.00 and assign it to  $x$ .
- Generate a random number between 0.00 and 1.00 and assign it to  $y$ .
- If  $(x^2 + y^2 \leq 1)$ , it means that this  $(x, y)$  coordinate lies *inside* the shaded quadrant.
- Repeat these steps  $N$  times, where  $N$  is sufficiently large number and then calculate the ratio of the points located inside the circle to the total number of generated points, i.e.,  $N$ . This ratio should approximate the area of one quadrant of this circle.
- Multiply the calculated ratio by 4 to find an approximation of the mathematical constant  $\pi$ .

Using the above procedure, draw a flowchart and write a C program to approximate the mathematical constant  $\pi$ .

Your program must read and validate the value of  $N$  (*a positive integer*) from the user at the beginning of execution.

Add as many inline comments as you can to make your program well documented and easy to be understood by anyone who reads it.

- Test your program using

$$N = 10, N = 100, N = 1\,000, N = 100\,000, N = 1\,000\,000, N = 10\,000\,000, \text{ and } N = 100\,000\,000.$$

- Loop inside your program to *recalculate* the value of  $\pi$  10 times using each of the above values of  $N$ . Calculate the mean (i.e., the average value) and standard deviation for results generated from the same value of  $N$ . Report all results (i.e., 70 values, 7 means and 7 standard deviations).
- Discuss your results.

$$\text{mean} = \frac{\sum_{i=1}^{i=n} X_i}{n}$$

$$\text{standard deviation} = \sqrt{\frac{\sum_{i=1}^{i=n} X_i^2 - (\sum_{i=1}^{i=n} X_i)^2}{n}}$$

## Program 2 (30 marks)

Draw a flowchart and write a C program that prints an  $n \times n$  *magic square* (a square arrangement of the numbers  $1, 2, \dots, n^2$  in which the sum of the elements in any row, column, or diagonal is the same). This program should generate a magic square of a specified size  $n$ , where the user will specify this size at run time. The size must be an *odd positive* integer number between 1 and 13 (to be validated by your program). You should store the magic square in a two-dimensional array.

Start by placing the number  $1$  in the middle of row  $0$ . Place each of the remaining numbers  $2, 3, \dots, n^2$  by moving *up one row* and *left one column*. Any attempt to go outside the bounds of the array should wrap around to the opposite side of the array. For example, if  $row = 0$ , instead of storing the next number in  $row - 1$ , we would store it in  $row = n - 1$  (the last row). If  $column = n - 1$ , instead of storing the next number in  $column = n$ , we would store it in  $column = 0$ . In addition, if a particular array element is already occupied (filled), put the number directly below the previously stored number. Here it is a sample output:

Enter size of magic square: 6			Enter size of magic square: 5				
Invalid size, try again...			17	24	1	8	15
Enter size of magic square: 3			23	5	7	14	16
8	1	6	4	6	13	20	22
3	5	7	10	12	19	21	3
4	9	2	11	18	25	2	9

In your program, you are **NOT allowed** to use the array notation, other than declaring the array. I.e., you can have `int magic_square[12][12];`

Yet, you are not allowed to use `magic_square[i][j]` to access the array elements in your program. Instead, you have to use the pointers notation to access the array elements.

Add as many inline comments in your program as you can to make it well documented and easy to be understood by anyone who reads it.

You should include enough *actual* test cases in your submission to demonstrate the functionality of your program, e.g., run your program with  $n = 1, 3, 4, 5, 11$ , and  $13$ .

### **Program 3 (35 marks)**

Draw a flowchart and write a C function that determines the smallest number of \$20, \$10, \$5, \$2, and \$1 bills/coins necessary to pay a dollar amount. The function prototype must be as follow:

```
void pay_amount(int dollars, int *twenties, int *tens, int *fives, int  
*toonies, int *lonnie);
```

where the dollar amount is represented by the `dollars` parameter. The `twenties` parameter points to a variable in which the function will store the number of \$20 bills required. The `tens`, `fives`, `toonies` and `lonnie` parameters are similar.

You should add as many inline comments as you can to make your code well documented and easy to be understood by anyone who reads it.

To test your function, write a program that asks the user to enter an integer value (a dollar amount), calls `pay_amount` to get the smallest number of bills/coins necessary to pay the amount, and then display the values returned by the function.

You should include enough *actual* test cases in your submission (i.e., a cut and paste from actual run to your program) to demonstrate the functionality of your program, to test various cases (at least 8 different cases).