

## CSE 1301 PRACTICAL SESSION 5

### Functions (10 marks)

By the end of this practical session, you should be able to write programs which call functions you have also written.

**Submission:** From now on, you must submit your pracs online via the following web page, as well as showing them to your demonstrator for marking. Your electronic submissions will be checked with an automatic cheating-detection program. Your electronic submissions are to be made by accessing the following website:

<https://www.csse.monash.edu.au/~pracwork/cgi-bin/student/submit.cgi>

**Coding style and documentation:** You are expected to document your programs and to use a standard, clear, and consistent coding style. Up to 2 of the marks for each practical class may be deducted for poor coding style and/or inadequate documentation.

#### **Preparation (to be completed before class)** **(2 marks)**

Write all the algorithms required for the questions below, and attempt the code for all the questions.

#### **PART 1** **(2 marks)**

Redo your algorithm for Part 1 (b), Prac 4 (Inverted Triangle) so that it calls a function to print the triangle. This function should be passed an integer value indicating the number of asterisks to print for a particular line of the triangle, there is no need to return a value. Code this new algorithm in C and save your program as **Prac5-1.c**.

Note: If you did not get your program working last Prac, you can use a solution available from [www.csse.monash.edu.au/courseware/cse1301/](http://www.csse.monash.edu.au/courseware/cse1301/)

#### **PART 2** **(3 marks)**

A proposed income tax system computes a person's tax based on the table below:

<i>Taxable income</i>	<i>Tax on this income</i>
1 to 6,300	None
6,301 to 11,700	13% for each 1 over 6,300
11,701 to 25,000	17% for each 1 over 11,700
25,001 to 40,000	22% for each 1 over 25,000
40,001 and over	28% for each 1 over 40,000

The **taxable income** is derived by deducting the deductible income from the total income:

*taxable income* = *total income* – *deductible income*, where

*deductible income* = (*personal relief* + *pension fund*)

The amount for personal relief is fixed at \$5000 for each individual.

Examples:

1. Total income = 20,000.75, pension fund = 3000.15  
 So, the taxable income =  $20000.75 - (5000.00 + 3000.15) = 12000.60$   
 Therefore the tax =  $(0.13 * (11700 - 6300)) + (0.17 * (12000.60 - 11700.00)) = 753.102$   
 The overall income tax rate =  $753.12/12000.60 = 6.28\%$
  2. Total income = 41,531.66, pension fund = 8053.99  
 So, the taxable income =  $41,531.66 - (5000.00 + 8053.99) = 28477.67$   
 Therefore the tax =  $(0.13 * (11700.00 - 6300.00)) + (0.17 * (25000.00 - 11700.00)) + (0.22 * (28477.67 - 25000.00)) = 3728.0874$   
 The overall income tax rate =  $3728.0874/28477.67 = 13.09\%$
- (a) Write an algorithm to read in a person's total income and pension fund contribution, and compute the tax to be paid and the overall income tax rate. Your algorithm should include at least two functions – one to compute the taxable income and the other to compute the tax.
  - (b) List a set of input data numbers that tests all possible executions of your algorithm.
  - (c) Implement your algorithm as a C program. Document and test your program. Save your program as **Prac5-2.c**.

**PART 3: Finding the root of a cubic polynomial****(3 marks)**

A quick test shows that the polynomial  $x^3 + 1.4x^2 - 0.4x - 0.8$  has a root between 0 and 1: the value at 0 is  $-0.8$ , and the value at 1 is  $1.2$ , so it has to have a value of zero somewhere in between. We can find the root of the polynomial by iteratively narrowing the interval in which we look for the root.

Our algorithm takes as input the left end point of the search interval ( $x_0$ ) and a parameter that determines the precision of the search ( $dx$ ). It works as follows:

1. probe the interval at several positions corresponding to the  $x$  values  $x = x_0 + i * dx$  for  $i = 0, 1, \dots, 10$
2. determine in which interval the function  $f(x)$  changes its sign (i.e. changes its value from negative to positive).
3. take the left end of this interval to be the left end  $x_0$  of the new search interval.
4. increase the precision from  $dx$  to  $0.1 * dx$ .
5. if the desired precision is not yet reached, repeat the algorithm from step 1 with the new values for  $x$  and  $dx$ .

Your task is to implement a C program that automates Step (1) of this algorithm. The rest of the algorithm will be executed manually.

(a) Implement an algorithm for Step (1) in a C program. Save your program as Prac5-3.c. The algorithm should take  $x_0$  and  $dx$  as input and print out the 11 pairs of values for the search points (the  $x$  value and the value  $f(x)$  of the polynomial at that point).

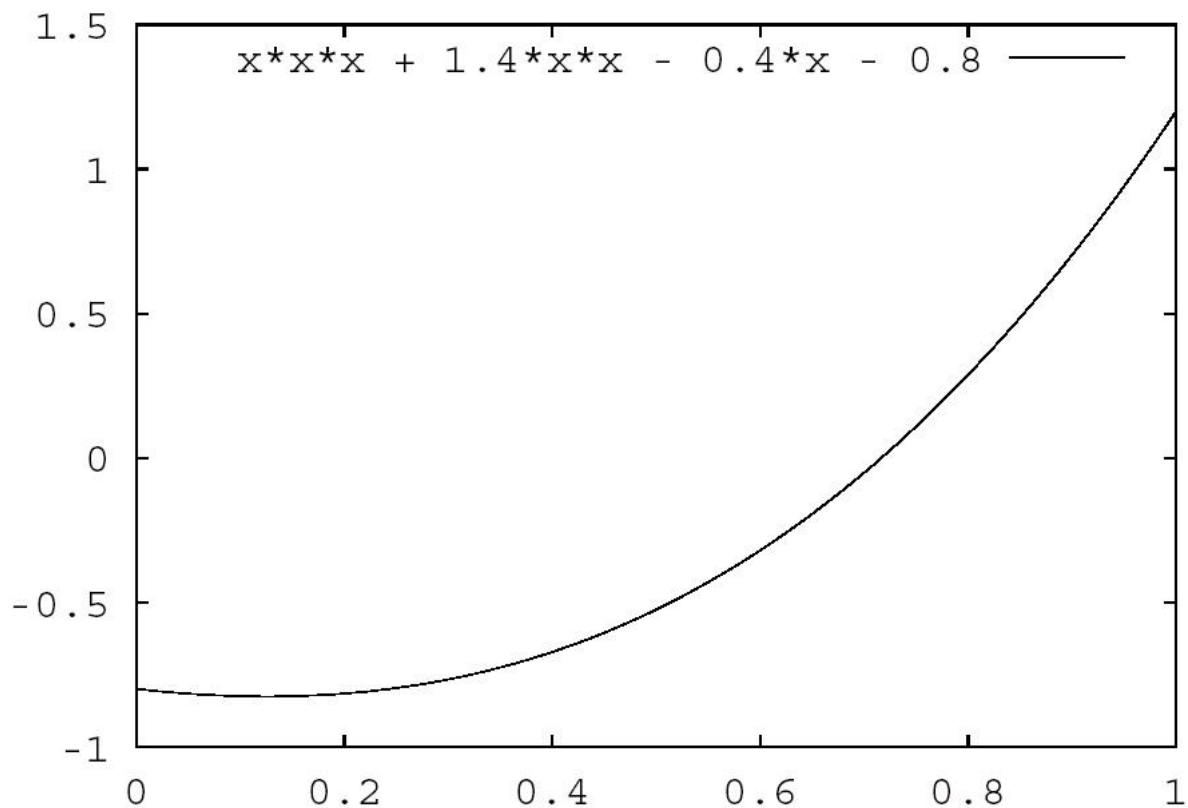
(b) Now execute the complete algorithm manually: Run your program 4 times, as follows, to achieve the desired accuracy.

The first time, the input is  $x_0 = 0.0$  and  $dx = 0.1$

The second time, the start value  $x_0$  depends on which subinterval the values change from negative to positive (observed from the previous run), and the increment is  $dx = 0.01$

The third time, do the same thing with an increment of  $dx=.001$

The fourth time, do the same thing with an increment of  $dx=.0001$  and round off to 3 places.



*Submission:*

Preparation: Show written preparation work to demonstrator at start of class for marking . If not done before class, it will not receive any marks, however you should still show it to your demonstrator during class to check your understanding.

PART 1, 2, 3: To be marked by the demonstrator during this class. No late submission.

### **Additional Component**

**(2 BONUS MARKS)**

Modify your program for Part 3 to eliminate the need for input by writing another function that does the four separate runs in a loop. You'll need to keep track of the value where the function changes from negative to positive.