

### CSE 1301 PRACTICAL SESSION 6 Pointers (15 marks)

The aim of this practical session is to give you experience using pointers, building upon the other programming skills you have learnt to date. You will have further practice using pointers. Parts 2 and 3 will also give you experience analysing an English language problem specifications, from which you will write first the algorithm and then the code.

**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 (Pointers) (2 marks)**

Write a program that contains two doubles, **x** and **y**, and a pointer to a double, **dblPtr**. Code the following steps, and use a print statement after each step to examine the values of all three variables (i.e. the contents of their memory locations), as well as the addresses of **x**, **y**, and the value of **\*dblPtr** (i.e., the contents of the memory location "pointed" to).

- (a) Read in values for **x** and **y** from the keyboard. Set **dblPtr** to point to **x**.
- (b) Set **\*dblPtr** to equal the current value of **y**. Add 1 to **\*dblPtr**.
- (c) Set **dblPtr** to point to **y**. Read in a new value for **x**.
- (d) Set **\*dblPtr** to be **4.56**, set **dblPtr** to **&x**. Set **x** to **9.0002** using **dblPtr**.

After variable creation	After step (a)	After step (b)	After step (c)	After step (d)
Address of x <input style="width: 100px; height: 20px;" type="text"/>	x <input style="width: 100px; height: 20px;" type="text"/>	x <input style="width: 100px; height: 20px;" type="text"/>	x <input style="width: 100px; height: 20px;" type="text"/>	x <input style="width: 100px; height: 20px;" type="text"/>
Address of y <input style="width: 100px; height: 20px;" type="text"/>	y <input style="width: 100px; height: 20px;" type="text"/>	y <input style="width: 100px; height: 20px;" type="text"/>	y <input style="width: 100px; height: 20px;" type="text"/>	y <input style="width: 100px; height: 20px;" type="text"/>
Address of dblPtr <input style="width: 100px; height: 20px;" type="text"/>	dblPtr <input style="width: 100px; height: 20px;" type="text"/> 	dblPtr <input style="width: 100px; height: 20px;" type="text"/> 	dblPtr <input style="width: 100px; height: 20px;" type="text"/> 	dblPtr <input style="width: 100px; height: 20px;" type="text"/> 
Value of *dblPtr <input style="width: 100px; height: 20px;" type="text"/>	? <input style="width: 100px; height: 20px; border: 1px dotted black;" type="text"/>	? <input style="width: 100px; height: 20px; border: 1px dotted black;" type="text"/>	? <input style="width: 100px; height: 20px; border: 1px dotted black;" type="text"/>	? <input style="width: 100px; height: 20px; border: 1px dotted black;" type="text"/>

**PART 2 (Pointers and Functions)****(6 marks)**

Following the steps below, write a C program to help a prospective borrower calculate the monthly payment for a loan. The program should prompt the user for the principal, the percent interest rate per year, and the duration of the time over which the loan will be repaid. The program should then print an information summary.

Banks and financial institutions use different formulas to calculate the monthly payment of a loan. For the purpose of this exercise, use the following simple formulas:

$$IM = \frac{IY}{12 \times 100}$$

$$P = (1 + IM)^D$$

$$Q = \frac{P}{P - 1}$$

$$MP = PR \times IM \times Q$$

where:

D: Duration (in months) over which loan will be repaid.

IY: Interest rate per year (as percentage)

IM: Interest rate per month (decimal)

PR: Principal (the amount of the loan)

MP: Monthly payment

**Note:** Test each of the functions below separately. You should also document your functions properly.

**(a) [0.5 marks]** Write a C function:

```
void inputInfo ( double *amountPtr, double *yearlyRatePtr,
                int *nmonthsPtr )
```

which prompts the user to input the amount of the loan, the interest rate per year (i.e., annual rate), and the number of months to amortize the loan. The input values should be stored in the variables pointed to by the pointers `amountPtr`, `yearlyRatePtr` and `nmonthsPtr`, respectively. For example:

```
Amount of loan (Principal)? 3000
Interest rate per year (in percent)? 9
Number of months to amortize loan? 12
```

**(b) [0.5 marks]** Write a C function

```
int NotValidInput ( double amount, double yearlyRate, int nmonths )
```

which returns 1 (true) if the input values are invalid (for example: the amount is not a positive number, and the `yearlyRate` is less than or equal to 0 or more than 100%). It returns 0 (false) otherwise.

**(c) [1 mark]** Write a C function:

```
double computeMonthlyRate ( double yearlyRate )
```

which calculates IM, the interest rate per month in decimal, given the interest rate per year

(yearlyRate). Use the formula given above.

**(d) [2 marks]** Write a C function:

```
double computeMonthlyPayment ( double amount, double monthlyRate,
                               int nmonths )
```

which calculates MP, the monthly payment, given the amount of the loan (amount), the interest rate per month (monthlyRate), and the number of months to amortize the loan (nmonths). Use the formula given above.

Rewrite this function so that instead of returning monthlyPayment , it is updated through a pointer. Use the following function header:

```
void computeMonthlyPayment ( double amount, double monthlyRate,
                             double *monthlyPaymentPtr, int nmonths )
```

**(e) [1 mark]** Write a C function:

```
void printInfo (double amount, double yearlyRate, int nmonths,
               double monthlyRate, double monthlyPayment )
```

which prints out an information summary about the loan. Here's a sample output:

```
Amount of loan:                3000.00
Interest rate/year (percent):  9.00
Interest rate/month (decimal): 0.007500
Number of months:              12
Monthly payment:               262.35
Total amount repaid:           3148.2
Total interest paid:           148.2
```

**(f) [1 mark]** Using the functions you have written in steps (a) to (e), write a C program which

- Prompts the user for the principal, the percent interest rate per year, and the number of months to amortize the loan.
- Checks if the input values are valid. If any of the input values are invalid, the program should end with a simple error message (e.g., "Invalid input").
- Calculates the interest rate per month.
- Calculates the monthly payment (remember to consider both versions of the function computeMonthlyPayment in your program).
- Prints the information summary about the loan.

Save your program as **Prac6-2.c**.

**PART 3:****(5 marks)**

It is a dark and stormy night. Our secret agent (007) is behind enemy lines at a fuel depot. He walks over to a cylindrical fuel tank that is 20 metres tall and 8 metres in diameter. He opens a 2-cm-diameter circular nozzle. He knows that the volume of the fuel leaving the tank is

$$\text{Volume lost} = \text{velocity} * (\text{area of the nozzle}) * \text{time}$$

And that

$$\text{Velocity} = 6.82 * (\text{height of fluid in the tank}) * 0.5$$

How long will it take to empty the tank?

While this simple problem should really be solved using calculus, we can use a stepwise simulation to approximate the precise solution that calculus would allow us to derive. This is done in the following way: we can calculate the volume lost over a short period of time, such as 60 seconds, and assume that the loss of fluid is constant. We can then subtract the volume from the tank and determine the new height of the fluid inside the tank at the end of the minute.

We can then calculate the loss for the next minute. This can be done over and over until the tank is dry. Print a table showing the elapsed time in seconds, the volume lost, and the height of the fluid. At the end convert the total elapsed seconds to minutes. The fluid height can be negative on the last line of the table.

Your program should include functions to calculate the volume lost and velocity respectively. Save your program as **Prac6-3.c**.

While for this particular problem there is no good reason not to apply calculus, many practical problems are too complex to be solved analytically and stepwise numerical simulation as illustrated above must be used to derive an approximate solution.

*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)**

Write an algorithm that, given a sequence of numbers, finds the *monotonically increasing* sub-sequence of consecutive numbers with the highest sum and prints its sum. For example, in the sequence

1	-3	3	4	6	5	9
---	----	---	---	---	---	---

the highest sum of a monotonically increasing sub-sequence is 14=5+9. You can assume that the input contains some positive numbers. Implement your algorithm in a C program, such that the core of the program is a function that receives as input the last number and the current number. It also receives as input a pointer to the sum so far of the subsequence currently under consideration, which it updates. Your program should contain a main loop that prompts for input and computes the highest sum using this function. It finishes and prints the sum when the user enters 0 (zero).